AD A007554

BIBLIOGRAPHIC DATA SHEET	1. Report No. USN-CCVS-74-VOL-06	2.	3. Recipient's Accession No.
4. Title and Subtitle: 1974 COBOL Compiler Validation System (X3.23-1974) ANS COBOL 1974 Indexed I-O Module Test Specifications			5. Report Date: 28 November 1974 6.
7. Author(s): Sec 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address: Software Development Division, Department of the Navy (ADPESO), Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address: ADPE Selection Office, Department of the Navy, Washington, D. C. 20376			13. Type of Report & Period Covered: Interim 14.

15. Supplementary Notes

16. Abstracts

This document represents one of 14 volumes comprising the preliminary documentation for the 1974 U.S. Navy COBOL Compiler Validation System (CCVS). The 1974 CCVS will consist of audit routines, their related data, and an execution reoutine (VP-Routine) which prepares the audit routine for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the result of the tests. The audit routines collectively contain all of the features of American National Standard Programs Language COBOL - X3.23-1974 (except for the ENTER statement of the Nucleus module) as specified in Federal Information Processing Standard (FIPS) 21-1.

17. Key Words and Document Analysis. 17a. Descriptors

COBOL
Validation
Software
Audit Routines
Verifying
Compilers
Standards
Programming Languages

CCS SUBJECT [illegible]

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement: Release Unlimited	19. Security Class (This Report): UNCLASSIFIED	21. No. of Pages: 116
	20. Security Class (This Page): UNCLASSIFIED	22. Price

DEPARTMENT OF THE NAVY
AUTOMATIC DATA PROCESSING EQUIPMENT SELECTION OFFICE
WASHINGTON, D.C. 20376

28 November 1974

Subject: 1974 COBOL Compiler Validation System - Indexed I-O Module

This document is one of a series produced by the Software Development Division, ADPE Selection Office, Department of the Navy. It contains specifications for audit routines which are to be part of the COBOL Compiler Validation System, and which will be used in testing COBOL compilers for conformance with one of the modules of the 1974 COBOL Standard as established by FIPS PUB 21-1. The reference document from which these specifications were produced is X3.23-1974, American National Standard Programming Language COBOL.

The 1974 COBOL Compiler Validation System will consist of audit routines, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes many tests and supporting procedures indicating the result of the tests. The audit routines collectively contain the features of Standard COBOL (except for the ENTER statement of the Nucleus module), as specified in FIPS PUB 21-1.

The validation of a compiler will determine the degree to which a compiler conforms to its language specification. The use of compilers that have attained a high degree of conformance with their respective language standards enhances program interchangeability within all ADP installations which use that particular programming language.

Thus, the purpose of producing a Validation System is to be able to test a COBOL compiler's adherence to the standard language syntax, and, where unambiguous, language semantics. The Validation System does not evaluate the implementation of a compiler nor its quantitative performance characteristics.

The purpose for releasing these specifications at this time is to provide adequate time for comments from interested bodies to influence the actual implementation of the audit routines. Review by impartial, technically competent individuals should expose any incorrect assumptions on the part of the specifications as well as oversights and obvious errors.

Persons interested in commenting on this set of specifications should forward their written comments to:

Director, Software Development Division
ADPE Selection Office
Department of the Navy
Washington, D. C. 20376

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O MODULE

Introduction

The Indexed I-O module provides a capability to access records of a mass storage file either random or sequential manner. The 3 access modes RANDOM, SEQUENTIAL and DYNAMIC allow for respective random, sequential and both sequential and random record retrieval. Logical file positioning for accessing the file sequentially is established by the OPEN and START statement.

Each record in an indexed file is uniquely identified by the value of one or more keys within that record. Duplicate key values are permitted for alternate keys and have a defined order of retrieval. There is one prime record key for an Indexed file and is used for inserting, updating and deleting records in the file, i.e., WRITE, REWRITE and DELETE verbs. The value for this key must, therefore, be unique.

This document is provided to aid in the evaluation and understanding of the Indexed I-O tests. The detail test specifications describe the language syntactical construct to be tested along with any semantic action that is expected. The nature of this module does not permit all the tests to be checked internally, therefore some visual checking of the source listing and diagnostics produced by the compiler is required.

The specifications for testing the Indexed I-O Module are divided into three sections:

a) Module Tests - This section contains a list of all Indexed I-O test programs and the number of tests contained in each program.

b) Matrix of COBOL elements - This section contains a matrix of the COBOL elements tested for Indexed I-O module and the test program in which it was tested.

c) Description of Test Program - This section contains a detail description of all the programs used to test the Indexed I-O module. Each program description provides an introduction defining the function or purpose of the test program followed by a description of each individual feature tested. For each feature tested, an objective is defined followed by as many supporting tests as necessary to adequately test that feature. A functional description along with its expected results is provided for each test.

The test programs take one of two formats:

1) Completely Self Contained - These are programs (run units) in which all files required for testing are created within that program and there is no file passing between test programs.

2) Test Set - This is a series of test programs in which files are passed from one program to the next. The tests are numbered in consecutive numerical order by test set.

The test programs depend heavily on the Nucleus Module and other features of the Indexed I-O Module for testing individual language features. To a lessor degree, the Table Handling Module is also used. The reference publication is American National Standard COBOL, X3.23-1974 (as adopted by FIPS PUB 21-1), Section VI, Indexed I-O Module.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O MODULE

Table of Contents

1974 CCVS SPECIFICATIONS

INDEXED I-O MODULE

Module Tests

	Program Name	Pass/Fail/Info	Visual	Total
Level 1	IX101	2	13	15
	IX102	11	9	20
	IX103	10	10	20
	IX104	12	11	23
Totals for Level 1		35	43	78

1974 CCVS SPECIFICATIONS

INDEXED I-O MODULE

Module Tests

	Program Name	Pass/Fail/Info	Visual	Total
Level 2	IX201	2	0	2
	IX202	11	1	12
	IX203	10	2	12
	IX204	12	3	15
	IX205	11	11	22
	IX206	11	6	17
	IX207	8	6	14
	IX208	30	30	60
	IX209	4	2	6
	IX210	27	0	27
	IX211	12	4	16
	IX212	9	2	11
	IX213	93	3	96
Totals for Level 2		240	70	310

fn = file-name
cn = condition-name
in = implementor-name
rn = record-name
id = identifier
i = integer
dn = data-name

MATRIX OF LANGUAGE ELEMENTS TESTED

INDEXED I-O MODULE

COBOL Language Elements	Programs in which language elements are tested																
	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
FILE-CONTROL (SELECT)																	
ASSIGN TO in	X		X		X						X	X					
ASSIGN in		X															
ORGANIZATION IS INDEXED	X		X								X			X			
;ORGANIZATION INDEXED		X															
ACCESS MODE IS SEQUENTL.	X		X		X								X	X			
;ACCESS MODE IS SEQUENTIAL										X							
ACCESS MODE IS RANDOM		X															
ACCESS SEQUENTIAL				X													
ACCESS MODE IS DYNAMIC						X	X	X	X					X	X		
;ACCESS MODE IS DYNAMIC												X					
ACCESS MODE DYNAMIC									X								
ACCESS DYNAMIC								X		X							
RECORD KEY IS dn	X	X	X		X	X	X				X	X		X			X
RECORD KEY dn									X	X							
RECORD IS dn													X				
RECORD dn				X				X		X							
RECORD KEY IS dn OF rn															X		
RECORD KEY IS dn OF dn IN dn															X		

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
ALTERNATE RECORD KEY IS dn									X						X		
ALTERNATE RECORD KEY IS dn WITH DUPLICATES													X	X		X	X
ALTERNATE RECORD KEY dn DUPLICATES											X						
ALTERNATE RECORD IS dn										X		X					X
ALTERNATE RECORD dn									X								
ALTERNATE RECORD dn WITH DUPLICATES											X						
ALTERNATE RECORD KEY IS dn OF dn																X	
RESERVE i AREA									X								
RESERVE i AREAS									X								
RESERVE i										X							
;RESERVE i AREAS										X							
FILE STATUS IS dn				X				X						X			X
FILE STATUS dn											X						
I-O-CONTROL (SAME)																	
RECORD FOR fn, fn									X								
RECORD fn fn									X								

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
FILE SECTION (FD file-name)																	
;BLOCK CONTAINS i RECORDS			X														
BLOCK CONTAINS i RECORDS	X																
BLOCK i RECORDS		X															
BLOCK i				X													
BLOCK CONTAINS i TO i RECORDS									X								
BLOCK i TO i RECORDS										X							
RECORD CONTAINS i CHARACTERS	X																
RECORD i			X														
RECORD i CHARACTERS		X															
LABEL RECORDS ARE OMITTED				X													
LABEL RECORD IS STANDARD	X																
LABEL RECORDS STANDARD		X															
LABEL RECORD STANDARD			X														
VALUE OF in IS literal	X																
VALUE OF in dn			X														
VALUE OF in IS literal, in IS dn										X							
VALUE OF in dn in dn											X						
DATA RECORD IS rn		X															
DATA RECORDS ARE rn				X													

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
;DATA RECORD rn		X															
DATA RECORDS rn				X													
PROCEDURE DIVISION (CLOSE) fn	X			X				X									
(DELETE) fn RECORD			X														
fn INVALID KEY							X										
(OPEN) INPUT fn			X														
OUTPUT fn				X													
I-O fn				X													
(READ) fn				X							X			X			
fn RECORD AT END											X			X			
fn AT END	X		X														X
fn NEXT RECORD AT END							X		X	X				X			
fn NEXT RECORD								X									
fn NEXT												X					
fn NEXT RECORD INTO id												X					
fn NEXT INTO id												X					
fn NEXT INTO id AT END												X					

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
fn RECORD INVALID KEY		X				X									X		
fn INVALID KEY									X	X							
fn KEY IS dn												X					
fn RECORD KEY IS dn OF dn IN dn															X		
fn INTO id KEY IS dn												X					
fn KEY IS dn OF dn																X	
fn RECORD KEY dn												X		X			
fn RECORD KEY dn INVALID												X					
fn RECORD KEY IS dn INVALID KEY												X					
(REWRITE) rn				X				X									
rn INVALID KEY		X				X								X			
(START) fn												X					
fn KEY EQUAL TO dn												X					
fn KEY IS EQUAL TO dn												X		X			
fn KEY IS EQUAL TO dn IN dn OF dn															X	X	
fn KEY IS EQUAL dn												X					
fn KEY IS = dn												X					
fn KEY IS GREATER THAN dn												X					

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
fn KEY IS GREATER THAN dn INVALID KEY																	X
fn KEY GREATER THAN dn												X					
fn KEY IS GREATER dn												X					
fn KEY IS > dn												X					
fn KEY IS > dn INVALID KEY																	X
fn KEY > dn												X					
fn KEY IS NOT LESS THAN dn												X					
fn KEY IS NOT LESS THAN dn INVALID KEY																	X
fn KEY IS NOT LESS dn												X					
fn KEY NOT LESS THAN dn												X					
fn KEY IS NOT < dn												X					
fn KEY IS EQUAL TO dn INVALID KEY												X		X			
fn KEY IS EQUAL TO dn INVALID												X					X
fn INVALID KEY												X		X	X		X
fn ;INVALID KEY												X					
fn KEY IS EQUAL TO dn ; INVALID KEY												X					
(USE AFTER) STANDARD EXCEPTION PROCEDURE ON fn				X				X						X		X	

	IX101	IX102	IX103	IX104	IX201	IX202	IX203	IX204	IX205	IX206	IX207	IX208	IX209	IX210	IX211	IX212	IX213
STANDARD ERROR PROCEDURE fn fn											X						
ERROR PROCEDURE fn fn												X	x				
(WRITE) rn	X			X				X									X
rn FROM id											X						
rn INVALID KEY	X				X										X		

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O MODULE

DESCRIPTION OF TEST PROGRAMS

INDEXED I-O Module - Level 1

IX1SET01 (Name of test set)
IX101 (Name of run unit)

GENERAL: This run unit is the first of a series which processes an Indexed file. The function of this program is to create an Indexed file sequentially (ACCESS MODE SEQUENTIAL) and verify that it was created as expected. The file is identified as 'IX-FS1' and is passed to subsequent run units for processing.

X-Card parameters which must be supplied for this program are:

X-24	Indexed file - for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	SELECT file-name-1 ORGANIZATION IS INDEXED RECORD KEY IS data- name-1 ACCESS MODE IS SEQUENTIAL See VI - 2.1.2.2 FD file-name-1 LABEL RECORDS STANDARD VALUE OF implementor- name IS literal See VI -3.3.2 WRITE record-name INVALID KEY See VI - 4.8.2	OBJECTIVE: This test creates an indexed file sequentially with the following characteristics: File size = 500 records Record size = 240 characters Blocking = 1 record Record key size = 29 characters Record key sequence = increasing numerical value by increments of 1 (1 thru 500) in positions 11 through 19 of the key. Key Positions 1 through 10 and 20 through 29 contain alphanumeric characters which are the same for each record key. See VI - 4.8.4(9) The WRITE statement
002	READ file-name-1 AT END See VI - 4.4.2	OBJECTIVE: This test reads the file created in INX-TEST-001 above and verifies the existance and accuracy of the 500 records created. The AT END phrase is used in the READ statement and it should be executed the 501st time the READ statement is executed. See VI - 4.4.4(9) The READ statement

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O Module - Level 1

IX1SET01 (Name of test set)
IX102 (Name of run unit)

GENERAL: The function of this run unit is to process an Indexed file randomly (ACCESS MODE IS RANDOM). The file used as input is that file created by IX101.

First the file is verified as to the existance and accuracy of the 500 records created in the first run unit. Secondly, records of the file are selectively updated; and thirdly, the accuracy of each record in the file is again verified.

X-Card parameters which must be supplied for the program are:

X-24	Indexed file for ASSIGN TO clause (same as IX101)
X-55	System printer
X-74	VALUE OF clause
X-75	Object of VALUE OF clause (same as IX101)
X-82	Source-Computer
X-83	Object-Computer.

//

003	SELECT file-name-1 ACCESS MODE IS RANDOM ;ORGANIZATION INDEXED RECORD KEY IS data-name-1 See VI - 2.1.2.2 READ file-name-1 INVALID KEY See VI - 4.2.2	OBJECTIVE: This test reads the file processed in the previous run unit and verifies the accuracy of the 500 records for use by this run unit. The numerical portion of the RECORD KEY is varied from 1 to 501 by increments of 1. The INVALID KEY path of the READ is expected to be taken upon encountering the 501st RECORD KEY value.
003.01		1. This test verifies that the RECORD KEY was set to the 501st key value when the INVALID KEY path of the READ statement was taken. See VI - 4.4.4(17) The READ Statement VI - 1.3.5 The INVALID KEY phrase.
003.02		2. This test verifies that the last record read was the record containing record number 500.
003.03		3. This test checks a counter which was incremented by 1 each time before the READ statement was executed. The counter is expected to be no greater than 501. A value greater than 501 indicates the INVALID KEY path of the READ was not taken. See VI - 4.4.4(17) the READ statement
003.04		4. This test verifies that each record of the file read, contained the appropriate record number in that record.
004	READ file-name-1 INVALID KEY REWRITE record-name INVALID KEY See VI - 4.5.2	OBJECTIVE: File-name-1 is opened as I-O and the file randomly read and updated on every fifth record. This is done by varying the numerical portion of the RECORD KEY from 5 to 505. All execution of the READ and REWRITE statements are expected to execute successfully except the last READ. When the RECORD KEY reaches 505 the next READ statement should take the INVALID KEY path. Each record updated will contain the value 'IC102' in the program-name field to indicate that the record was updated in this program. Also the update-number field of the record will be incremented by 1.

004.01	1. This test checks to see that the INVALID KEY path of the READ was taken when the RECORD KEY reached 505. A value greater than 505 causes the test to fail. The program flow then passes to the next test.
004.02	2. Each time an INVALID KEY path is taken on the READ statement during the reading of the file a counter is incremented. The counter is expected to be zero for this test.
004.03	3. Each time an INVALID KEY path is taken on the REWRITE statement during the updating of this file, a counter is incremented. The counter is expected to be zero for this test. See VI - 4.5.4(8) THE REWRITE statement
005	OBJECTIVE: This test reads the file updated in INX-TEST-004 and verifies that the appropriate updates were made. This is done by varying the numerical portion of the RECORD KEY from 500 to 00. The INVALID KEY of the READ statement is expected to be taken only when the RECORD KEY reaches Zero. There should be 100 updated records and 400 nonupdated records in the file.
005.01	1. Before each READ statement is executed, a counter is decremented by 1. The counter is expected to be no less than Zero. A value of less than Zero indicates that the INVALID KEY path of the READ was not taker.
005.02	2. Each time a record was read that was not updated by this program a counter was incremented by 1. The counter is expected to equal 400.
005.03	3. Each time a record was read that was updated by this program a counter was incremented by 1. The counter is expected to be equal to 100.
005.04	4. Each time the INVALID KEY path was taken on the READ statement, a counter was incremented. The counter is expected to be no greater than 1. The only INVALID KEY that is expected

is when the RECORD KEY reaches ZERO.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O Module - Level 1

IX1SET01 (Name of test set)
IX103 (Name of run unit)

GENERAL: This run unit is the third of a series. The function of this run unit is to process the file sequentially (ACCESS MODE IS SEQUENTIAL). The file used is that resulting from IX102.

First, the file is verified for accuracy of its 500 records. Secondly, records of the file are selectively deleted and thirdly the accuracy of each record in the file is again verified.

X-Card parameters which must be supplied for this program are:

X-24	Indexed file - ASSIGN TO clauses (same as IX102)
X-55	System printer
X-74	VALUE OF clause
X-75	Object of VALUE OF clause (same as IX102)
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
006	SELECT file-name-1 ORGANIZATION IS INDEXED RECORD KEY IS data-name-1	OBJECTIVE: This test reads the file processed by the previous run unit and verifies the accuracy of each of the 500 records for use by this run unit. The file is read sequentially. The RECORD KEY should at all times contain the index of the record previously read. There are 500 records in the file. The absence of the ACCESS clause is treated as though the SEQUENTIAL had been specified.
	READ file-name-1 AT END	INFORMATION TEST: The contents of the RECORD KEY following the opening of the file is not defined by the specifications. The current record pointer is updated to point to the first record in the file prior to execution of the first read statement.
006.01		1. Before each READ statement is executed a counter is incremented by 1. The counter is to be no greater than 501. A value greater than 501 indicates that the AT END path was not taken following the last record of the file.
006.02		2. Each time a record was read that was not updated by the previous run unit, a counter was incremented by 1. The counter should equal 400.
006.03		3. Each time a record was read that was updated by the previous run unit, a counter was incremented by 1. The counter is expected to equal 100.
006.04		4. Each record read is verified to make sure it contains the record number reflecting the current record. If the record number does not match, the counter is incremented. All records of the file are expected to match.
007	READ file-name-1 AT END DELETE file-name-1 RECORD See VI - 4.2.2 VI - 4.2.3(1)	OBJECTIVE: The file is opened as I-O and read with every fourth record being deleted. Once a record is deleted it should no longer be available for future accesses. There will be 125 records deleted from the file.

Before the record is deleted, the program-name field is updated to reflect the program 'IX103' in which the record was deleted. Also the update-number field will contain a value of 99. Test INX-TEST-008 checks to see that there are none of these records in the file.

007.01

1. A counter is incremented before the READ statement is executed. The AT END path of the READ statement is expected to be taken when the counter is equal 501 indicating all the records of the file have been read.

007.02

2. This test checks a counter to verify that the DELETE statement was executed the appropriate number of times. The counter should be equal to 125.
   See VI - 4.2.4(2) The DELETE statement

008 READ file-name-1
AT END

OBJECTIVE: This test reads the file updated in INX-TEST-007 and verifies that the records were in fact deleted from the file. The only records which should be remaining are those which were not deleted. There should be 375 records in the file.

008.01

1. This test checks the counter used to count the number of times the READ statement was executed until the AT END path was taken. The counter should equal 376.

008.02

2. This test checks the counter used to count the number of deleted records found in the file. This count should be equal to zero.
   See VI - 4.2.4(4) The DELETE statement

008.03

3. This test checks a counter which counts the number of records in which the actual RECORD KEY contents match the expected RECORD KEY contents. Each record read should provide a match and thus the counter should equal 375.

008.04

4. This test checks for the number of

records in which the record retrieved was the record expected to be provided as a result of the READ. Each record read should provide a match and thus the counter should equal 375.

IX104 (Name of run unit)

GENERAL: This run unit tests the syntactical constructs and semantic actions associated with the following elements:

(1) FILE STATUS
(2) USE AFTER EXCEPTION using file-name
(3) READ
(4) WRITE
(5) REWRITE
(6) RECORD KEY
(7) ACCESS

This program creates an Indexed file sequentially (ACCESS MODE SEQUENTIAL) and then updates selective records of the file. The FILE STATUS contents are captured and tested for accuracy for each OPEN, CLOSE, READ and REWRITE statement used. The READ, WRITE and REWRITE statements are used without the appropriate AT END or INVALID KEY phrases. The omission of these phrases are permitted if an applicable USE procedure has been specified.

X-Cards which must be supplied for this program are:

X-24	Indexed file - for ASSIGN TO clause
X-55	System printer
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
001	OBJECTIVE: Test permissible syntactical constructs for ACCESS, RECORD KEY, FILE STATUS, USE and WRITE language features. SELECT file-name-1 ACCESS SEQUENTIAL RECORD data-name-1 FILE STATUS IS data-name-2 See VI - 2.1.2.1 USE AFTER STANDARD EXCEPTION PROCEDURE ON file-name-1 See VI - 4.7.2 WRITE record-name See VI - 4.8.2 See VI - 4.8.3(3)	OBJECTIVE: An Indexed file is created sequentially with the following characteristics: File size = 500 records RECORD KEY size = 29 characters Positions 1 thru 10 alphanumeric characters (fixed) Positions 11 thru 19 numeric characters Positions 20 thru 29 alphanumeric characters (fixed) RECORD KEY sequence = 1 thru 500 in the numerical portion of the key Blocking = 1 Record size = 240 characters The file is created without the INVALID KEY phrase for the WRITE statement. Omission of this phrase is permitted if an applicable USE procedure has been specified. The contents of data-name-1 (FILE STATUS) following each execution of the OPEN, CLOSE, and WRITE statement are verified for accuracy.
001.01		1. This test checks for any execution of the USE procedures. Any INVALID KEY condition, in which this phrase was not specified for the WRITE, should cause the USE procedure to be executed. For this test an INVALID KEY condition is not expected to occur. See VI - -4.7.4(1) The USE statement VI - -1.3.5.3 The INVALD KEY phrase.
001.02		2. This test checks the last record written to verify that it was record number 500.
001.03		3. This test checks the contents of the FILE STATUS (data-name-2) following the OPEN OUTPUT statement. The result of the

		OPEN is expected to be successful and thus contain the contents '00'. See VI - 4.3.4(1), (2), (12) The OPEN statement and VI - 1.3.4 I-O Status
001.04		4. This test checks the FILE STATUS (data-name-2) following each WRITE statement. All WRITEs are expected to be successful, thus the contents of data-name-2 should be equal to '00'. See VI - 4.8.4(5) The WRITE statement. See VI - 1.3.4 I-O Status.
001.05		5. This test checks the FILE STATUS data-item (data-name-2) following execution of the CLOSE statement. The close of the file is expected to be successful, thus the contents of data-name-2 should be equal to '00'. See VI - 1.3.4 The I-O Status.
002	OBJECTIVE: Test the permissible syntactical constructs for the READ and REWRITE statements without the appropriate AT END or INVALID KEY phrase. READ file-name-1 See VI - 4.4.2 REWRITE record-name See VI - 4.5.2	OBJECTIVE: Read and update the file created in INX-TEST-009. When the READ and REWRITE statements are used without the AT END or INVALID KEY phrase respectively, any such condition occurring should cause the appropriate USE procedure to be executed. The contents of the FILE STATUS data-item should be updated for each execution of the OPEN I-O, READ, REWRITE and CLOSE statements.
002.01		1. The file is opened with the I-O option and contents of data-name-2 are captured for later reference. This test checks the number of times the USE procedure was executed while the file was being updated. The USE procedure is expected to be executed only once i.e., when the end-of-file condition occurs for the READ statement. All executions of of the REWRITE statement are expected to be successful. If any

REWRITEs were unsuccessful the USE procedure should have been executed.

See VI - 4.4.4(10)c The READ statement.
See VI - 4.7.4(1) The USE statement.

002.02

2. This test checks the number of times the READ statement was executed. It should have executed 501 times. The last time should have found an end-of-file condition and thus cause the USE procedure to be executed.

See VI - 4.4.4(10)c The READ statement.

002.03

3. This test checks the contents of the FILE STATUS data-item (data-name-2) following the OPEN I-O statement The open is expected to be successful thus data-name-2 should be equal to '00'.

See VI - 4.3.4(1), (2), (10) The OPEN statement; and VI - 1.3.4 I-O Status.

002.04

4. This test reads all the records in the file and checks the contents of the FILE STATUS data-item (data-name-2) following the end-of-file condition. The data-item should have been updated to reflect the AT END condition. thus data-name-2 should equal '10'.

See VI - 1.3.4 I-O Status; and VI - 4.4.4(10)a The READ statement.

002.05

5. This test captures the contents of data-name-2 during execution of the USE procedures triggered by the AT END condition of the READ statement. The execution sequence should be such that the FILE STATUS data-item is updated before any applicable USE procedure is executed. The contents of data-name-2 should be equal to '10'.

See VI - 1.3.4 I-O Status.

002.06

6. This test captures the contents of the FILE STATUS data-item (data-name-2) during execution of the USE

procedure. The AT END condition should update data-name-2 to '10' and should occur after the 500th record is read.

See VI - 4.4.4(3) The READ statement.

002.07

7. This test checks the contents of the FILE STATUS data-item (data-name-2) following execution of the CLOSE statement. The CLOSE statement is expected to be successful and thus update data-name-2 to equal '00'.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O Module - Level 2

IX2SET01 (Name of test set)
IX201 (Name of run unit)

GENERAL: This run unit is the first of a series which processes an Indexed file. The function of this progrc is to create an Indexed file sequentially (ACCESS MODE SEQUENTIAL) and verify that it was created as expected. The file is identified as 'IX-FS1' and is passed to subsequent run units for processing.

X-Card parameters which must be supplied for this program are:

X-24	Indexed file - for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	SELECT file-name-1 ORGANIZATION IS INDEXED RECORD KEY IS data- name-1 ACCESS MODE IS SEQUENTIAL See VI - 2.1.2.2 FD file-name-1 LABEL RECORDS STANDARD VALUE OF implementor- name IS literal See VI -3.3.2 WRITE record-name INVALID KEY See VI - 4.8.2	OBJECTIVE: This test creates an indexed file sequentially with the following characteristics: File size = 500 records Record size = 240 characters Blocking = 1 record Record key size = 29 characters Record key sequence = increasing numerical value by increments of 1 (1 thru 500) in positions 11 through 19 of the key. Key Positions 1 through 10 and 20 through 29 contain alphanumeric characters which are the same for each record key. See VI - 4.8.4(9) The WRITE statement
002	READ file-name-1 AT END See VI - 4.4.2	OBJECTIVE: This test reads the file created in INX-TEST-001 above and verifies the existance and accuracy of the 500 records created. The AT END phrase is used in the READ statement and it should be executed the 501st time the READ statement is executed. See VI - 4.4.4(9) The READ statement

INDEXED I-O Module - Level 2

IX2SET01 (Name of test set)
IX202 (Name of run unit)

GENERAL: The function of this run unit is to process an Indexed file randomly using the ACCESS MODE IS DYNAMIC clause. The file used as input is that file created by IX201.

First the file is verified as to the existance and accuracy of the 500 records created in the first run unit. Secondly, records of the file are selectively updat d; and thirdly, the accuracy of each record in the file is again verified.

X-Card parameters which must be supplied for the program are:

X-24	Indexed file for ASSIGN TO clause (same as IX201)
X-55	System printer
X-74	VALUE OF clause
X-75	Object of VALUE OF clause (same as IX201)
X-82	Source-Computer
X-83	Object-Computer.

003	SELECT file-name-1 ACCESS MODE IS DYNAMIC ORGANIZATION IS INDEXED RECORD KEY IS data-name-1 READ file-name-1 INVALID KEY	OBJECTIVE: This test randomly reads the file processed in the previous run unit and verifies the accuracy of the 500 records for use by this run unit. The numerical portion of the RECORD KEY is varied from 1 to 501 by increments of 1. The INVALID KEY path of the READ is expected to be taken upon encountering the 501st RECORD KEY value. The file is being read under the ACCESS MODE IS DYNAMIC clause; however, was created with the ACCESS MODE IS SEQUENTIAL specification. See VI - 2.1.2.4(7) The SELECT clause.
003.01		1. This test verifies that the RECORD KEY was set to the 501st key value when the INVALID KEY path of the READ statement was taken. See VI - 4.4.4(17) The READ Statement VI - 1.3.5 The INVALID KEY phrase.
003.02		2. This test verifies that the last record read was the record containing record number 500.
003.03		3. This test checks a counter which was incremented by 1 each time before the READ statement was executed. The counter is expected to be no greater than 501. A value greater than 501 indicates the INVALID KEY path of the READ was not taken. See VI - 4.4.4(17) the READ statement
003.04		4. This test verifies that each record of the file contained the appropriate record number in that record.
004	READ file-name-1 INVALID KEY REWRITE record-name INVALID KEY See VI - 4.5.2	OBJECTIVE: File-name-1 is opened as I-O and the file randomly read and updated on every fifth record. This is done by varying the numerical portion of the RECORD KEY from 5 to 505. All READ and REWRITE statements are expected to execute successfully except the last READ. When the RECORD KEY reaches 505 the next READ statement should take the INVALID KEY path.

Each record updated will contain the value 'IX202' in the program-name field to indicate that the record was updated in this program. Also the update-number field of the record will be incremented by 1.

004.01

1. This test checks to see that the INVALID KEY path of the READ was taken when the RECORD KEY reached 505. A value greater than 505 causes the test to fail. The program flow then passes to the next test.

004.02

2. Each time an INVALID KEY path is taken when reading the file, a counter is incremented. The counter is expected to be zero for this test.

004.03

3. Each time an INVALID KEY path is taken on the REWRITE statement while updating the file, a counter is incremented. The counter is expected to be zero for this test.
See VI - 4.5.4(9) THE REWRITE statement

005

OBJECTIVE: This test reads the file updated in INX-TEST-004 and verifies that the appropriate updates were made. This is done by varying the numerical portion of the RECORD KEY from 500 to 00. The INVALID KEY of the READ statement is expected to be taken only when the RECORD KEY reaches zero. There should be 100 updated records and 400 nonupdated records in the file.

005.01

1. Before each READ statement is executed, a counter is decremented by 1. The counter is expected to be no less than Zero. A value of less than Zero indicates that the INVALID KEY path of the READ was not taken.

005.02

2. Each time a record was read that was not updated by this program a counter was incremented by 1. The counter is expected to equal 400.

005.03

3. Each time a record was read that was updated by this program a counter was incremented by 1. The counter is expected to be equal to 100.

005.04

4. Each time the INVALID KEY path was taken on the READ statement, a counter was incremented. The counter is expected to be no greater than 1. The only INVALID KEY that is expected is when the RECORD KEY reaches zero.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O Module - Level 2

IX2SET01 (Name of test set)
IX203 (Name of run unit)

GENERAL: This run unit is the third of a series. The function of this run unit is to process the file sequentially using the ACCESS MODE IS DYNAMIC clause. The file used is that resulting from IX202.

First, the file is verified for accuracy of its 500 records. Secondly, records of the file are selectively deleted and thirdly the accuracy of each record in the file is again verified.

X-Card parameters which must be supplied for this program are:

X-24	Indexed file - ASSIGN TO clauses (same as IX202)
X-55	System printer
X-74	VALUE OF clause
X-75	Object of VALUE OF clause (same as IX202)
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
006	SELECT file-name-1 ACCESS MODE IS DYNAMIC ORGANIZATION IS INDEXED RECORD KEY IS data- name-1	OBJECTIVE: This test reads the file processed by the previous run unit and verifies the accuracy of each of the 500 records for use by this run unit. The file is read sequentially. The RECORD KEY should at all times contain the index of the record previously read. There are 500 records in the file.
	READ file-name-1 NEXT RECORD AT END	INFORMATION TEST: The contents of the RECORD KEY following the opening of the file is not defined by the specifications. The current record pointer is updated to point to the first record in the file prior to execution of the first read statement.
006.01		1. Before each READ statement is executed a counter is incremented by 1. The counter is to be no greater than 501. A value greater than 501 indicates that the AT END path was not taken after the the last record of the file was read.
006.02		2. Each time a record was read that was not updated by the previous run unit, a counter was incremented by 1. The counter should equal 400.
006.03		3. Each time a record was read that was updated by the previous run unit, a counter was incremented by 1. The counter is expected to equal 100.
006.04		4. Each record read is verified to made sure it contains the record number reflecting the current record. If the record number does not match, the counter is incremented. All records of the file are expected to match.
007	READ file-name-1 NEXT RECORD AT END DELETE file-name-1 INVALID KEY See VI - 4.2.2 See VI - 4.2.3(2)	OBJECTIVE: The file is opened as I-O and read with every fourth record being deleted. Once a record is deleted it should no longer be available for future accesses. There will be 125 records deleted from the file. Before the record is deleted, the program-name field is updated to reflect the

		program 'IX203' in which the record was deleted. Also the update-number field will contain a value of 99. Test INX-TEST-008 checks to see that there are none of these records in the file.
007.01		1. A counter is incremented before the READ statement is executed. The AT END path of the READ statement is expected to be taken when the counter is equal 501 indicating all the records of the file have been read.
007.02		2. This test checks a counter to verify that the DELETE statement was executed the appropriate number of times. The counter should be equal to 125. See VI - 4.2.4(3) The DELETE statement
008	READ file-name-1 NEXT RECORD AT END	OBJECTIVE: This test reads the file updated in INX-TEST-007 and verifies that the records were in fact deleted from the file. The only records which should be remaining in the file are those which were not deleted. There should be 375 records remaining in the file.
008.01		1. This test checks the counter used to count the number of times the READ statement was executed. The AT END path is expected to be taken when the counter is equal to 376.
008.02		2. This test checks the counter used to count the number of deleted records found in the file. This count should be equal to zero. See VI - 4.2.4(4) The DELETE statement
008.03		3. This test checks a counter which counts the number of records in which the actual RECORD KEY contents match the expected RECORD KEY contents. The RECORD KEY should at all times contain the index of the record previously read. Each record read should provide a match and thus the counter should equal 375.
008.04		4. This test checks for the number of

records in which the record retrieved was the record expected to be provided as a result of the READ. Each record read should provide a match and thus the counter should equal 375.

IX204 (Name of run unit)

GENERAL: This run unit tests the syntactical constructs and semantic actions associated with the following elements:

(1) FILE STATUS
(2) USE AFTER ERROR PROCEDURE ON file-name
(3) READ
(4) WRITE
(5) REWRITE
(6) RECORD KEY
(7) ACCESS

This program creates an Indexed file sequentially (ACCESS MODE DYNAMIC) and then updates selective records of the file. The FILE STATUS contents are captured and tested for accuracy for each OPEN, CLOSE, READ and REWRITE statement used. The READ, WRITE and REWRITE statements are used without the appropriate AT END or INVALID KEY phrases. The omission of these phrases are permitted if an applicable USE procedure has been specified.

X-Cards which must be supplied for this program are:

X-24	Indexed file - for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of the VALUE OF clause
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
001	OBJECTIVE: Test permissible syntactical constructs for ACCESS, RECORD KEY, FILE STATUS, USE and WRITE language entries. SELECT file-name-1 ACCESS DYNAMIC RECORD data-name-1 FILE STATUS IS data-name-2 See VI - 2.1.2.1 USE AFTER STANDARD EXCEPTION PROCEDURE ON file-name-1 See VI - 4.7.2 WRITE record-name See VI - 4.8.2 See VI - 4.8.3(3)	OBJECTIVE: An Indexed file is created sequentially under the DYNAMIC ACCESS MODE and contain the following file characteristics: File size = 500 records RECORD KEY size = 29 characters Positions 1 thru 10 alphanumeric characters (fixed) Positions 11 thru 19 numeric characters Positions 20 thru 29 alphanumeric characters (fixed) RECORD KEY sequence = 1 thru 500 in the numerical portion of the key Blocking = 1 Record size = 240 characters The file is created without the INVALID KEY phrase for the WRITE statement. Omission of this phrase is permitted if an applicable USE procedure has been specified. Data contents of data-name-2 (FILE STATUS) following each execution of the OPEN, CLOSE, and WRITE statement are verified for accuracy.
001.01		1. This test checks for any execution of the USE procedures. Any INVALID KEY condition, in which this phrase was not specified for the WRITE, should cause the USE procedure to be executed. For this test an INVALID KEY condition is not expected to occur. See VI - 4.7.4(1) The USE statement VI - 4.8.3.3 The INVALID KEY phrase.
001.02		2. This test checks the last record written to verify that it was record number 500.
001.03		3. This test checks the contents of the FILE STATUS (data-name-2) following the OPEN OUTPUT statement. The result of the

		OPEN is expected to be successful and thus contain the contents '00'. See VI - 4.3.4(1), (2), (12) The OPEN statement and VI - 1.3.4 I-O Status
001.04		4. This test checks the FILE STATUS (data-name-2) following each WRITE statement. All WRITEs are expected to be successful, thus the contents of data-name-2 should be equal to '00'. See VI - 4.8.4(5) The WRITE statement. See VI - 1.3.4 I-O Status.
001.05		5. This test checks the FILE STATUS data item (data-name-2) following execution of the CLOSE statement. The close of the file is expected to be successful, thus the contents of data-name-2 should be equal to '00'. See VI - 1.3.4 The I-O Status.
002	OBJECTIVE: Test the permissible syntactical constructs for the READ and REWRITE statements without the appropriate AT END or INVALID KEY phrase. READ file-name-1 NEXT RECORD See VI - 4.4.2 REWRITE record-name See VI - 4.5.2	OBJECTIVE: Read and update the file created in IX-TEST-009. When the READ and REWRITE statements are used without the AT END or INVALID KEY phrase respectively, any such condition occurring should cause the appropriate USE procedure to be executed. The contents of the FILE STATUS data-item should be updated for each execution of the READ, REWRITE, OPEN I-O and CLOSE statement.
002.01		1. The file is opened as I-O and the contents of data-name-2 captured for later reference. This test checks the number of times the USE procedure was executed while the file was being updated. The USE procedure is expected to be executed only once i.e., when the end-of-file condition occurs for the READ statement. All executions of of the REWRITE statement are ex-

pected to be successful. If any REWRITEs were unsuccessful the USE procedure should have been executed.

See VI - 4.4.4(10)c The READ statement.
See VI - 4.7.4(1) The USE statement.

002.02

2. This test checks the number of times the READ statement was executed. It should have executed 501 times. The last time should have found an end-of-file condition and thus cause the USE procedure to be executed.

See VI - 4.4.4(10)c The READ statement.

002.03

3. This test checks the contents of the FILE STATUS data-item (data-name-2) following the OPEN I-O statement The open is expected to be successful and thus data-name-2 should be equal to '00'.

See VI - 4.3.4(1), (2), (10) The OPEN statement; and VI - 1.3.4 I-O Status.

002.04

4. This test reads all the records in the file and checks the contents of the FILE STATUS data-item (data-name-2) following the end-of-file condition. The data-item should have been updated to reflect the AT END condition. thus data-name-2 should equal '10'.

See VI - 1.3.4 I-O Status; and VI - 4.4.4(10)a The READ statement.

002.05

5. This test captures the contents of data-name-2 during execution of the USE procedures triggered by the AT END condition of the READ statement. The execution sequence should be such that the FILE STATUS data-item is updated before any applicable USE procedure is executed. The contents of data-name-2 should be equal to '10'.

37

See VI - 1.3.4 I-O Status.

002.06

6. This test captures the contents of the FILE STATUS data-item (data-

name-2) during execution of the USE procedure. The AT END condition should update data-name-2 to '10' and should occur after the 500th record is read.

See VI - 4.4.4(9) The READ statement.

002.C7

7. This test checks the contents of the FILE STATUS data-item (data-name-2) following execution of the CLOSE statement. The CLOSE statement is expected to be successful and thus update data-name-2 to equal '00'.

IX205 (Name of run unit)

GENERAL: The function of this run unit is to test the permissible syntactical constructs of COBOL elements associated with Level 2 Level 2 of INDEXED I-O. The elements tested in this routine are:

(1) ACCESS MODE DYNAMIC;
(2) ALTERNATE RECORD KEY without the duplicates option;
(3) RESERVE clause;
(4) SAME clause;
(5) BLOCK CONTAINS integer-1 TO integer-2 clause;
(6) VALUE OF implementor-name.

Each element tested will be exercised semantically by this routine. Files are created and accessed using the ACCESS MODE IS DYNAMIC.

X-Cards which must be supplied for this program are:

X-24	Indexed file-1 for ASSIGN TO clause
X-25	Indexed file-2 for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause
X-76	Object of VALUE OF clause for file-2
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	OBJECTIVE: Test the full syntactical constructs associated with the RESERVE, ACCESS and ALTERNATE RECORD clause.	OBJECTIVE: Create a file using the ACCESS MODE IS DYNAMIC and verify that the records can be retrieved Randomly as well as Sequentially.
001.01	SELECT file-name-1 RESERVE integer-1 area ACCESS MODE IS DYNAMIC RECORD KEY IS data-name-1 ALTERNATE RECORD KEY IS data-name-2 See VI - 2.1.2.2	1. This test creates a file of 200 records. The RECORD KEY is 10 positions in length and established in numerical sequence by increments of 1. The ALTERNATE RECORD KEY is 10 positions in length and established in inverse numerical sequence by increments of 1. See VI - 4.8.4(13) The WRITE Statement.
001.02	READ file-name-1 NEXT RECORD AT END	2. This test opens the file as INPUT and verifies that the records can be accessed sequentially. The test reads 25 records. See VI - 2.1.2.4(7).
001.03	READ file-name-1 INVALID KEY	3. This test verifies that the records can be accessed randomly. The test reads 10 records. See VI - 2.1.2.4(7).
001.04		4. Read 25 records sequentially using the alternate key and verify that the records were retrieved in the correct sequence. The START statement is used to establish the key of reference.
002	OBJECTIVE: Test use of the various syntactical constructs associated with the RESERVE, ACCESS, ALTERNATE RECORD, BLOCK CONTAINS and VALUE OF clauses.	OBJECTIVE: Create a file in the DYNAMIC Access Mode and verify that the records can be retrieved Randomly as well as Sequentially. The file created is expected to contain the appropriate label record identification and blocking characteristics as defined by the VALUE OF and BLOCK clause respectively.

002.01	SELECT file-name-2 RESERVE integer-1 AREAS ACCESS MODE DYNAMIC RECORD KEY data-name-3 ALTERNATE RECORD data-name-4 See VI - 2.1.2.2 VALUE OF implementor-name data-name-5 BLOCK CONTAINS integer-1 TO integer-2 RECORDS See VI - 2.1.3.2	1. This test creates a file of 200 records. The RECORD KEY is 10 positions in length and established in numerical sequence by increments of 1. The ALTERNATE RECORD KEY is 10 positions in length and established in inverse numerical sequence by increments of 1. The file blocking is defined as 5 to 25 records per block. See VI - 4.8.4(13) The WRITE statement.
002.02	READ file-name-2 NEXT RECORD AT END	2. This test opens the file as INPUT and verifies that the records can be accessed sequentially. The test reads 25 records. See VI - 2.1.1.4(7).
002.03	READ file-name-2 INVALID KEY	3. This test verifies that the record can be accessed randomly. The test reads 10 records. See VI - 2.1.2.4(7).
002.04		4. Read 25 records sequentially using the alternate key and verify that the records were retrieved in the correct sequence. The START statement is used to establish the key of reference.
003	OBJECTIVE: Test use of the various syntactical constructs associated with the SAME clause.	OBJECTIVE: Verify that files specified in the SAME RECORD clause share the same record area.
003.01	SAME RECORD file-name-1 file-name-2 See VI - 2.1.3.2	1. File-name-1 and file-name-2 are opened as INPUT. A record is read sequentially from file-name-1 and its contents verified.
003.02		2. A record is read sequentially from file-name-2 and a test is made to verify that an AT END path was not taken on either the read of file-name-1 or file-name-2.

003.03

3. The shared record areas of file-name-1 and file-name-2 are tested. The data contents of the most recently read file (file-name-2) is expected. Contents are verified by using record references associated with file-name-1.

See VI - 2.1.3.4(4).

INDEXED I-O

LEVEL 2

IX206 (Name of run unit)

GENERAL: The function of this routine is to test the permissible syntactical constructs of COBOL elements associated with Level 2 of INDEXED I-O The elements tested in this routine are.

(1) ACCESS MODE DYNAMIC
(2) ALTERNATE RECORD KEY without the duplicates option
(3) RESERVE clause
(4) SAME clause
(5) BLOCK CONTAINS integer-1 TO integer-2 clause
(6) VALUE OF implementor-name series.

Each element tested will be exercised semantically by this routine. Files are created and accessed in the DYNAMIC Access Mode.

X-Cards which must be supplied for this program are:

X-24	Indexed file-1 for ASSIGN TO clause
X-25	Indexed file-2 for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	OBJECTIVE: Test the syntactical constructs associated with the RESERVE, ACCESS, ALTERNATE RECORD, BLOCK and VALUE OF clauses	OBJECTIVE: Create a file in the DYNAMIC mode and verify that the records can be retrieved randomly, as well as sequentially. The file created is expected to contain the appropriate label record identificaton and blocking characteristics as defined by the VALUE OF and BLOCK clause respectively.
001.01	SELECT file-name-1 RESERVE integer-1 ACCESS DYNAMIC RECORD KEY IS data-name-1 ALTERNATE RECORD IS data-name-2 See VI - 2.1.2.2 VALUE OF implementor-name data-name-3 implementor-name data-name-4 BLOCK integer-1 TO integer-2 RECORDS See VI - 3.3.2	1. This test creates a file of 200 records. The RECORD KEY is 10 positions in length and established in numerical sequence order by increments of 1. The ALTERNATE RECORD KEY is 10 positions in length and established in inverse numerical sequence by increments of 1. The file blocking is defined as 10 to 20 records per block. See VI - 4.8.4(13) The WRITE statement.
001.02	READ file-name-1 NEXT RECORD AT END	2. This test opens the file as INPUT and verifies that the records can be accessed sequentially. The test reads 25 records. See VI - 2.1.2.4(7).
001.03	READ file-name-1 INVALID KEY	3. This test verifies that the records can be accessed randomly. The test reads 10 records. See VI - 2.1.2.4(7).
001.04		4. Read 25 records sequentially using the alternate key and verify that the records were retrieved in the correct sequence. The START statement is used to establish the key of reference.
002	OBJECTIVE: Test use of the abbreviated syn-	OBJECTIVE: Create a file in the SEQUENTIAL ACCESS MODE and verify that the records can

	tactical constructs associated with the RESERVE, ACCESS, ALTERNATE RECORD and VALUE OF clauses	be retrieved sequentially on both the prime and alternate record keys. The file created is expected to contain the appropriate label record identificaton as defined by the VALUE OF clause.
002.01	SELECT file-name-2 ;RESERVE integer-1 AREAS ;ACCESS MODE IS SEQUENTIAL RECORD KEY data-name-3 ALTERNATE RECORD KEY data-name-4 See VI - 2.1.2.2 VALUE OF implementor-name IS literal, implementor-name IS data-name-5 See VI - 3.3.2	1. This test creates a file of 200 records. The RECORD KEY is 10 positions in length and established in numerical sequence order by increments of 1. The ALTERNATE RECORD KEY is 10 positions in length and established in inverse numerical sequence by increments of 1. See VI - 4.8.4(13) The WRITE statement.
002.02	READ file-name-2 NEXT RECORD AT END	2. This test opens the file as INPUT and verifies that the records can be accessed sequentially. The test reads 25 records.
002.03	READ file-name-2 INVALID KEY	3. Read 25 records sequentially using the alternate key and verify that the records were retrieved in the correct sequence. The START statement is used to establish the key of reference.
003	OBJECTIVE: Test use of the permissible syntactical constructs associated with the SAME clause	OBJECTIVE: Verify that files specified in the SAME RECORD clause share the same record area. File-name-1 and file-name-2 specify different access modes.
003.01	SAME RECORD FOR file-name-1, file-name-2 See VI - 2.1.3.2	1. File-name-1 and file-name-2 are opened as input. A record is read sequentially from file-name-1 and its contents verified.
003.02		2. A record is read sequentially from file-name-2 and a test is made to verify that an AT END path was not taken on either the read of file-name-1 or file-name-2.
003.03		3. The record read from file-name-2 is verified as a record of that file.

003.04

4. The shared record areas of file-name-1 and file-name-2 are tested. The data contents of the most recently read file (file-name-2) is expected. Contents are verified by using record references associated with file-name-1.

See VI - 2.1.2.4(4), VI - 2.1.3.3(7).

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O
LEVEL 2

IX207 (Name of run unit)

GENERAL: The function of this routine is to test the permissible syntactical constructs of COBOL elements associated with Level 2 of INDEXED I-O. The elements tested in this routine are:

(1) Ordering of clauses in file-control-entry;
(2) ALTERNATE RECORD KEY with the duplicates option;
(3) USE AFTER STANDARD EXCEPTION file-name-1, file-name-2;
(4) FILE STATUS.

Each element tested will be exercised semantically by this routine. Files are created and accessed in the SEQUENTIAL ACCESS MODE.

X-Cards which must be supplied for this program are:

X-24	Indexed file-1 for ASSIGN TO clause
X-25	Indexed file-2 for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause
X-76	Object of VALUE OF clause for file-2
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
001	OBJECTIVE: Test use of the abbreviated syntactical constructs associated with the ALTERNATE RECORD, FILE STATUS and positioning of clauses in the file-control-entry. The SELECT clause must be specified first in the file-control entry and the remaining clauses may appear in any order. SELECT file-name-1 ACCESS MODE IS SEQUENTIAL ALTERNATE RECORD data-name-1 WITH DUPLICATES FILE STATUS data-name-2 RECORD KEY IS data-name-3 ORGANIZATION IS INDEXED ASSIGN TO implementor-name See VI - 2.1.2.2: VI - 2.1.2.3(1)	OBJECTIVE: This test creates and processes an indexed file sequentially (ACCESS MODE IS SEQUENTIAL) utilizing the syntactical constructs specified in the file-control-entry. The file will contain the following characteristics: File size = 300 records Record size = 240 characters Record Key size = 29 records Alternate Key size = 29 characters Record Key sequence = Positions 1 thru 5: decreasing numerical value by increments of 1 (300 thru 1). Positions 6 thru 29: any combination of alphanumeric characters which are the same for each record Key. Alternate Key sequence = Positions 1 thru 24: any combination of alphanumeric characters which are the same for each alternate Key. Positions 25 thru 29: increasing numerical value by increments of 1 with every 50th and 51st record Key a duplicate key. The DUPLICATES option of the ALTERNATE clause should permit non-unique alternate Key values to be processed. Retrieval of these records should be in the order in which they were written.
001.01	WRITE record-name FROM identifier INVALID KEY	1. This test creates the file defined above. A count is kept of the number of times the WRITE statement is executed. Upon encountering the 300th record a test is made to verify that there were no unsuccessful WRITEs (INVALID KEY path). The DUPLICATES option of the ALTERNATE clause should permit non-unique alternate Key values to be processed. See VI - 4.8.4(14): VI - 4.8.4(15)a. The WRITE statement.

001.02	READ file-name-1 RECORD AT END	2. Open file-name-1 as INPUT and read 110 records of the file sequentially verifying the contents of each record. The OPEN should position the current record pointer to the first record of the file and establish the prime record key as the current key of reference. See VI - 4.3.4(9), The OPEN statement.
001.03		3. Twenty records of file-name-1 are read sequentially using the ALTERNATE KEY as the key of reference. Each record is verified. The START statement is used to establish data-name-1 as the key of reference. See VI - 4.4.4(2)a, The READ statement.
001.04	READ file-name-1 RECORD AT END	4. This test reads 60 records of file-name-1 and verifies that the duplicate key can be accessed. Each record read is verified. The ALTERNATE KEY is used as the key of reference. See VI - 4.4.4(14), The READ statement.
001.05		5. This test reads the file until the AT END path has been taken. The FILE STATUS data-item (data-name-2) should contain the value '10'. See VI - 2.1.2.4, The FILE clause.
002	OBJECTIVE: Test use of the abbreviated syntactical constructs associated with the ALTERNATE, and USE entries SELECT file-name-2 ALTERNATE RECORD KEY data-name-4 DUPLICATES RECORD KEY IS data-name-5 See VI - 2.1.2.2 USE AFTER STANDARD EXCEPTION PROCEDURE file-name-1, file-	OBJECTIVE: This test creates and processes an indexed file sequentially utilizing the syntactical constructs specified in the file-control-entry. The USE procedures should be executed for each occurrence in which the AT END phrase of the READ is not specified and an end-of-file condition occurs. The file characteristics are the same as that specified in INX-TEST-001 above. When no ACCESS clause is specified, the ACCESS MODE IS SEQUENTIAL option is assumed.

name-2

See VI - 4.7.2

002.01	WRITE record-name FROM identifier INVALID KEY	1. This test creates the file defined above. Upon encountering the 300th record a test is made to verify that there were no INVALID KEY paths taken during creation of the file.
002.02	READ file-name-1 READ file-name-2	2. This test READs file-name-1 and file-name-2 until an end-of-file condition occurs. The appropriate USE procedure should be executed upon encountering the 301st read of each file.
002.03	READ file-name-2 RECORD AT END	3. This test reads 120 records verifying that the records can be retrieved via the alternate key containing duplicate key values. Each record is verified as for the appropriate data contents. The START statement is used to establish data-name-4 as the key of reference.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O - Level 2

IX208 (Name of run unit)

GENERAL: The function of this routine is to test the permissible syntactical constructs of COBOL elements associated with level 2 of the INDEXED I-O Module. The elements tested in this routine are:

(1) READ statement;
(2) START statement;
(3) USE statement.

Each element tested will be exercised semantically by this routine.

X-Cards which must be supplied for this program are:

X-24	Indexed file-1 for ASSIGN TO clause
X-25	Indexed file-2 for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause for file-1
X-76	Object of VALUE OF clause for file-2
X-82	Source-Computer
X-83	Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
001	OBJECTIVE: Test use of the abbreviated syntactical constructs associated with the USE statement	OBJECTIVE: The objective is to create two files for testing the READ and START statements. The characteristics defined in the SELECT clause are used for file-name-1 and file-name-2 respectively. Each file will contain the following file characteristics:
	SELECT file-name-1 ALTERNATE RECORD KEY IS data-name-1 RECORD KEY IS data-name-2 ASSIGN TO imple- mentor-name ;ACCESS MODE IS DYNAMIC	File size = 300 records Record size = 240 characters Blocking = 1 record RECORD KEY size = 10 characters ALTERNATE KEY size = 10 characters RECORD KEY sequence = increasing numerical value by increments of 1 (1 thru 300).
	SELECT file-name-2 ALTERNATE RECORD KEY IS data-name-3 RECORD KEY IS data-name-4 ACCESS MODE IS SEQUENTIAL ASSIGN TO imple- mentor-name	ALTERNATE KEY sequence = decreasing numerical values (300 thru 1).
	USE AFTER ERROR PROCEDURE file-name-1 file- name-2	
	See VI - 4.7.2	
	WRITE record-name INVALID KEY	
001.01	WRITE record-name INVALID KEY	1. This test creates file-name-1 using the file characteristics defined above. When the file has been created, a test is made to verify that the WRITE statement was executed 300 times and that there were no INVALID KEY paths taken.
001.02		2. This test creates file-name-2 using the file characteristics defined above. When the file has been created, a test is made to verify that the WRITE statement was executed 300 times and there were no

INVALID KEY paths taken.

002	OBJECTIVE: Test use of the following syntactical construct variations for the READ statement. See VI - 4.4.2 In tests INX-TEST-002.01 thru INX-TEST-002.04 the NEXT phrase of the READ statement is used to retrieve records sequentially from a file specified in the DYNAMIC Access Mode. See VI - 4.4.3(5) In tests INX-TEST-002.05 thru INX-TEST-002.09 the READ statement is used to retrieve records randomly from a file specified in the DYNAMIC Access Mode. See VI - 4.4.3(6)	OBJECTIVE: Use each of the specified READ statements to retrieve records from the file and verify that the expected record was provided.
002.01	READ file-name-1 NEXT	1. This test reads 10 records and verifies that the expected records were provided.
002.02	READ file-name-1 NEXT RECORD INTO identifier	2. This test reads 10 records and verifies that the expected records were provided. The contents of the identifier are also verified for accuracy.
002.03	READ file-name-1 NEXT INTO identifier	3. This test reads 10 records and verifies that the expected records were provided. The contents of the identifier are also verified for accuracy.
002.04	READ file-name-1 NEXT INTO identifier AT END	4. This test reads the file until an end-of-file condition occurs. The AT END path should be taken upon encountering the 301st read.

002.05	READ file-name-1 KEY IS data-name-2	5. This test reads 10 records and verifies that the expected records were provided. The prime record key is used to retrieve the record.
002.06	READ file-name-1 INTO identifier KEY IS data-name-2	6. This test reads 10 records and verifies that the expected record was provided. The "identifier" is also verified for accuracy. The prime record key is used to retrieve the record.
002.07	READ file-name-1 RECORD KEY data-name-1	7. This test reads 10 records and verifies that the expected records were provided. The alternate record key is used to retrieve the record.
002.08	READ file-name-1 RECORD KEY data-name-2 INVALID	8. This test reads 10 records. Each key value provided does not match any key value existing in the file. The INVALID KEY path should be taken for each read. The prime record key is used.
002.09	READ file-name-1 RECORD KEY IS data-name-1 INVALID KEY	9. This test reads 10 records. Each key value provided does not match any key value existing in the file. The INVALID KEY path should be taken for each read. The alternate record key is used.
003	OBJECTIVE: Test use of the following syntactical construct variations for the START statement See VI - 4.6.2 The tests INX-TEST-003.08 through INX-TEST-003.16 will use the alternate key as the key of reference. See VI - 4.6.3(5) Tests INX-TEST-003.01 through INX-TEST-003.07 and INX-TEST-003.17 thru INX-TEST-003.19 will use	OBJECTIVE: Read the file sequentially and verify that each of the specified START statements correctly positioned the current record pointer.

the prime record as the Key of reference

See VI - 4.6.3(5)

Tests INX-TEST-003.01 through INX-TEST-003.14 will use file-name-2 which is designated as ACCESS MODE IS SEQUENTIAL

See VI - 4.6.3(2)

Tests INX-TEST-003.15 through INX-TEST-003.19 will use file-name-1 which is designated as ACCESS MODE IS DYNAMIC

See VI - 4.6.3(2)

003.01	START file-name-2	1. Following execuon of the START, 10 records are read sequentially and verified for accuracy. When the KEY phrase is not specified, the primary record key (data-name-4) is used as the key of reference.
003.02	START file-name-2 KEY EQUAL TO data-name-4	2. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records to be accessed. The primary key is used as the key of reference (data-name-4).
003.03	START file-name-2 KEY IS EQUAL TO data-name-4	3. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records to be accessed. The primary key is used as the key of reference (data-name-4).
003.04	START file-name-2 KEY IS EQUAL data-name-4	4. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records to be accessed. The primary key is used as the key of reference (data-name-4).
003.05	START file-name-2 KEY IS = data-name-4	5. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records

		to be accessed. The primary key is used as the key of reference (data-name-4).
003.06	START file-name-2 KEY IS GREATER THAN data-name-4	6. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records to be accessed. The primary key is used as the key of reference (data-name-4).
003.07	START file-name-2 KEY GREATER THAN data-name-4	7. Read 10 file-name-2 records sequentially and verify that the START statement permitted the correct records to be accessed. The primary key is used as the key of reference (data-name-4).
003.08	START file-name-2 KEY IS GREATER data-name-3	8. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.09	START file-name-2 KEY IS > data-name-3	9. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.10	START file-name-2 KEY > data-name-3	10. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.11	START file-name-2 KEY IS NOT LESS THAN data-name-3	11. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.12	START file-name-2 KEY IS NOT LESS data-name-3	12. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.13	START file-name-2 KEY NOT LESS THAN data-name-3	13. Read 10 file-name-2 records sequentially and verify the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.14	START file-name-2 KEY IS NOT <	14. Read 10 file-name-2 records sequentially and verify the START statement permitted

	data-name-3	the correct record to be accessed. The alternate record key is used as the key of reference (data-name-3).
003.15	START file-name-1 KEY IS EQUAL TO data-name-1 INVALID KEY	15. Read 10 records sequentially via the "READ file-name-1 NEXT RECORD" statement and verify that the START statement permitted the correct record to be accessed. The alternate record key is used as the key of reference (data-name-1).
003.16	START file-name-1 KEY IS EQUAL TO data-name-1 INVALID	16. Establish a value in data-name-1 equivalent to the 301st logical record of the file. The INVALID KEY path is expected to be executed. The alternate record key is used as the key of reference (data-name-1).
003.17	START file-name-1 INVALID KEY	17. Establish a value in the prime record key (data-name-2) equivalent to the 301st logical record of the file. The INVALID KEY path is expected to be executed.
003.18	START file-name-1 ;INVALID KEY	18. Establish a value in the prime record key (data-name-2) equivalent to the 301st logical record of the file. The INVALID KEY path is expected to be executed.
003.19	START file-name-1 KEY IS EQUAL TO data-name-1; INVALID KEY	19. Establish a value in the prime record key (data-name-1) equivalent to the 300th logical record of the file. Establish a value in the alternate record key (data-name-1) to a value that does not match any other alternate key value in the file. The INVALID KEY path is expected to be executed.

IX2SET02 (Name of test set)
IX209 (Name of run unit)

GENERAL: This run unit is the first of a series which processes 2 Indexed files. The function of this program is to create 2 Indexed files sequentialy (ACCESS MODE SEQUENTIAL) and verify that is was created as expected. The files are passed to subsequent run units for processing.

X-Card parameters which must be supplied for this program are:

X-24 Indexed file - for ASSIGN TO clause (passed to IX210)
X-25 Indexed file - for ASSIGN TO clause (passed to IX210)
X-55 System printer
X-74 VALUE OF implementor-name
X-75 Object of VALUE OF clause for file X-24 above
X-76 Object of VALUE OF clause for file X-25 above
X-82 Source-Computer
X-83 Object-Computer.

TEST IMX-TEST-	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	SELECT file-name-1 ORGANIZATION IS INDEXED RECORD KEY IS data-name-1 ALTERNATE RECORD KEY IS data- name-2 WITH DUPLICATES ACCESS MODE IS SEQUENTIAL See VI - 2.1.2.2 FD file-name-1 LABEL RECORDS STANDARD VALUE OF implementor- name IS literal See VI - 3.3.2 WRITE record-name INVALID KEY See VI - 4.8.2	OBJECTIVE: This test creates an indexed file sequentially with the following characteristics: File size = 300 records Record size = 240 characters Blocking = 1 record ALTERNATE KEY size = 29 characters ALTERNATE sequence = increasing numerical value by increments of 1 (1 thru 300) in positions 1 thru 10 of the key. Key positions 11 thru 29 contain alphanumeric characters which are the same for each record key. RECORD KEY size = 10 characters RECORD KEY sequence = increasing numerical value by increments of 2 (2 thru 600). See VI - 4.8.4(9) The WRITE statement.
002	READ file-name-1 AT END See VI - 4.4.2	OBJECTIVE: This test reads file-name-1 created in INX-TEST-001 above and verifies the existance and accuracy of the 300 records created. The AT END phrase is used in the READ statement and it should be executed the 301st time the READ statement is executed. See VI - 4.4.4(9) The READ statement.
003	SELECT file-name-2 ORGANIZATION IS INDEXED RECORD KEY IS data-name-3 ALTERNATE RECORD	OBJECTIVE: This test creates an indexed file sequentially with the following characteristics: File size = 300 records Record size = 240 characters

	KEY IS data-name-4 WITH DUPLICATES ACCESS MODE IS SEQUENTIAL See VI - 2.1.2.2 FD file-name-2 LABEL RECORDS STANDARD VALUE OF implementor-name IS literal See VI - 3.3.2 WRITE record-name INVALID KEY See VI - 4.8.2	Blocking = 1 record ALTERNATE KEY size = 29 characters ALTERNATE sequence = increasing numerical value by increments o 1 (1 thru 300) in positions 1 thru 10 of the key. Key positions 11 thru 29 contain alphanumeric characters which are the same for each record key. RECORD KEY size = 10 characters RECORD KEY sequence = increasing numerical value by increments of 2 (2 thru 600). See VI - 4.8.4(9) The WRITE statement.
004	READ file-name-2 AT END See VI - 4.4.2	OBJECTIVE: This test reads file-name-2 created in INX-TEST-003 above and verifies the existance and accuracy of the 300 records created. The AT END phrase is used in the READ statement and it should be executed the 301st time the READ statement is executed. See VI - 4.4.4(9) The READ statement.

1974 CCVS TEST SPECIFICATIONS

INDEXED I-O Module - Level 2

IX2SET02 (Name of test set)
IX210 (Name of run unit)

GENERAL: This is the second run unit in the test set. This run unit processes 2 files, one has been designated as ACCESS MODE IS DYNAMIC and the second is designated as ACCESS MODE IS SEQUENTIAL. The function of this run unit is to test level 2 feature associated with the READ statement. The features tested are as follows.

(1) Verify that the current record pointer can be positioned properly for sequential READs.

(2) Verify that the records read, which contain duplicate key values, are retrieved in the sequence in which they were written to the set.

(3) Use of a random READ, for a file designated as DYNAMIC, to establish the current record pointer for subsequent sequential READs.

(4) Verify that the FILE STATUS data-item was updated to reflect the presence of duplicate keys.

X-Card parameters which must be supplied for this program are:

X-24 Indexed file - for ASSIGN TO clause (From IX209)
X-25 Indexed file - for ASSIGN TO clause (From IX209)
X-55 System printer
X-74 VALUE OF implementor-name
X-75 Object of VALUE OF clause for X-24
X-76 Object of VALUE OF clause for X-25
X-82 Source-Computer
X-83 Object-Computer.

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTIONS
005	SELECT file-name-1 ORGANIZATION IS INDEXED RECORD KEY IS data-name-1 ALTERNATE RECORD KEY IS data- name-2 WITH DUPLICATES ACCESS MODE IS SEQUENTIAL FILE STATUS IS data-name-3 SELECT file-name-2 ORGANIZATION IS INDEXED RECORD KEY IS data-name-4 ALTERNATE RECORD KEY IS data- name-5 WITH DUPLICATES ACCESS MODE IS DYNAMIC FILE STATUS IS data-name-6 USE AFTER STANDARD ERROR PROCEDURE ON file-name-2	OBJECTIVE: Read and verify the 300 records for each file created in the previous run unit for use by this run unit. The file is opened as I-O for reading and verification. In addition every 49th and 55th record is updated via the REWRITE such that the alternate key values are changed. This updating is used in later tests for verifying the correct record retrieval sequence of modified alternate key values and duplicate key values. NOTE: All explicit record references, unless otherwise noted, will be in terms of the RECORD KEY values.
005.01	READ file-name-1 RECORD AT END REWRITE record-name INVALID KEY	1. This test reads file-name-2 and verifies each record for accuracy. On every 49th read the alternate key value (data-name-2) is made equal to the alternate key value of the 50th record, alternate key value decreased by 1, and the record rewritten. On every 55th read the alternate key value is increased by 4 and the record rewritten. A check is made to verify that all the 300 records were processed.
005.02		2. This test verifies that all READs, and REWRITEs were successful, i.e., no INVALID KEYs or unexpected AT END path was taken.

005.03	READ file-name-2 NEXT RECORD AT END REWRITE record-name INVALID KEY	3. This test reads the file and verifies each record for accuracy. On every 49th read the alternate key value (data-name-2) is made equal to the alternate key value of the 50th record (the alternate key value is decreased by 1) and the record rewritten. On every 55th read the alternate key value is increased by 4 and the record rewritten. A check is made to verify that all the 300 records were processed.
005.04		4. This test verifies that all READs, and REWRITEs were successful, i.e., no INVALID KEYs or unexpected AT END path was taken.
006		OBJECTIVE: Verify that the appropriate records can be retrieved sequentially when the current record pointer is updated or modified. The file used is designated as the ACCESS MODE IS SEQUENTIAL.
006.01	READ file-name-1 RECORD AT END	1. The file is opened as input to establish the current record pointer to the first record currently existing within the file and establish the prime record key as the key of reference. The read statement is executed and the record verified as being the first one in the file.
006.02	START file-name-1 KEY IS EQUAL TO data-name-2 READ file-name-1 RECORD AT END	2. The START statement is used to alter the current key of reference from the prime record key to the alternate record key. A sequential read (format 1) is used to verify that the current record pointer has been updated to the new key of reference. See VI - 4.4.4(2)a The READ statement.
006.03	START file-name-1 KEY IS EQUAL TO data-name-2 READ file-name-1 RECORD AT END	3. The START statement is used to modify the current record pointer to another starting position in the file. The key of reference is the same as the previous read test (INX-TEST-006.02). The READ is expected to retrieve the record as positioned by the START statement. See VI - 4.4.4(2)a The READ statement.
007		OBJECTIVE: Verify that the appropriate records can be retrieved sequentially when the current record pointer is updated or modified. The file used is that designated in the

DYNAMIC Access Mode.

007.01	READ file-name-2 NEXT RECORD AT END	1. The file is opened as input to establish the current record pointer to the first record currently existing within the file and establish the prime record key as the key of reference. The read statement is executed and the record verified as being the first one in the file.
007.02	READ file-name-2 NEXT RECORD AT END	2. The START statement is used to alter the current key of reference from the prime record key to the alternate record key. A sequential read (format 1) is used to verify that the current record pointer has been updated to the new key of reference. See VI - 4.4.4(2)a The READ statement.
007.03	READ file-name-2 NEXT RECORD AT END	3. The START statement is used to modify the current record pointer to another starting position in the file. The key of reference will be the same as the previous read test (INX TEST-007.03). The READ is expected to retrieve the record positioned by the START statement. See VI - 4.4.4(2)a The READ statement.
008	READ file-name-2 NEXT RECORD AT END	OBJECTIVE: Test the record retrieval sequence for alternate keys which have duplicate key values when the sequential READ is used. The retrieval sequence is expected to be in the order in which they were created by the WRITE or updated by the REWRITE. The file being processed is in the DYNAMIC mode. The keys and associated values, in RECORD KEY sequence, are as follows with the numbers in parenthesis indicating the sequence in which the duplicate key values were created.

RECORD KEY (data-name-4)	ALTERNATE KEY (data-name-5)	
47	253	
48	252	
49	250 (3)	update via REWRITE
50	250 (1)	created via WRITE
51	250 (2)	created via WRITE
52	248	

53	247	
54	246	
55	249	- updated via REWRITE
56	244	

NOTE: All explicit record references, unless otherwise noted, will be in terms of the RECORD KEY values.

008.01 START file-name-2
INVALID KEY

START file-name-2
KEY IS EQUAL TO
data-name-5
INVALID KEY

1. Open file-name-2 as input and load data-name-4 with a Key value for the 110th record (alternate Key value 190).
The START statement is used to establish the current record pointer firstly for the prime record Key as the Key of reference and secondly for the alternate Key as the Key of reference. There are 61 records read read from the file. The 61st record should have the same Key value (Key value 250) as 49th and 51st records and be the first record written via the WRITE when the file was created.

See VI - 4.4.4(14) The READ statement.

008.02

2. Read the next record. This should be the 51st record created. The record contains the same Key values as the 50th and 49th records and should be the second record written via the WRITE statement when the file was created.

See VI - 4.4.4(14) The READ statement.

008.03

3. Read the next record. This should be the 49th record. The record contains the same Key values as the 50th and 51st records and should be the record updated via the REWRITE statement.

See VI - 4.4.4(14) The READ statement.

009

OBJECTIVE: Test the record retrieval sequence for alternate keys which have duplicate values when the sequential READ statement is used. The retrieval sequence is expected to be in the order in which they were created by the WRITE or REWRITE. The file being processed is in the SEQUENTIAL mode. The keys and associated values in their respective sequence are with the numbers in parenthesis indicating the sequence in which the duplicate key

values were created:

RECORD KEY (data-name-1)	ALTERNATE KEY (data-name-2)
47	253
48	252
49	250 (3) update via REWRITE
50	250 (1) created via WRITE
51	250 (2) created via WRITE
52	248
53	247
54	246
55	249 - updated via REWRITE
56	244

NOTE: All explicit record references unless otherwise noted, will be in terms of the RECORD KEY values.

009.01 START file-name-1
INVALID KEY

START file-name-1
KEY IS EQUAL TO
data-name-2
INVALID KEY

1. Open file-name-1 as input and load data-name-1 with the key value for the 110th record (alternate key value 190). The START is used to establish the current record pointer firstly for the prime record key and secondly for the alternate key as the key of reference. Sixty-one records are read (key values are in inverse sequence). This should be the 50th record, have the same key value (key value 250) as the 49th and 51st records and be the first record written via the WRITE when the file was created.

See VI - 4.4.4(14) The READ statement.

009.02

2. Read the next record. This should be the 51st record created. The record contains the same key value as the 50th and 49th records and should be the second record written via the WRITE statement when the file was created.

See VI - 4.4.4(14) The READ statement.

009.03

3. Read the next record. This should be the 49th record. The record contains the same key value as the 50th and 51st records and should be the record updated via the

REWRITE statement.

See VI - 4.4.4(14) The READ statement.

010		OBJECTIVE: Test the record retrieval sequence for alternate keys which have duplicate values when the random READ statement is used. The file being processed is in the DYNAMIC mode. A sample of the keys and associated values, in RECORD KEY sequence , are shown in INX-TEST-008 above.
010.01	READ file-name-2 RECORD KEY IS data-name-5 INVALID KEY	1. Before the record is read, the ALTERNATE KEY data-item (data-name-5) is loaded with key value 250. Following the read the record retrieved is checked. This test expects the 50th record created via the WRITE statement to be provided. See VI - 4.4.4(17) The READ statement.
010.02		2. This test determines whether or not the INVALID KEY path was taken as a result of the READ in INX-TEST-010.01 above. This test does not expect an INVALID KEY condition. See VI - 4.4.4(17) The READ statement.
010.03		3. This test checks the FILE STATUS data-item (data-name-6) following the read in INX-TEST-010.01. This test expects the READ to be successful and STATUS KEY to reflect the presence of duplicate keys thus data-name-6 should contain the contents "01". See VI - 1.3.4.1 Status Key 1 and VI - 1.3.4.2 Status Key 2.
010.04	READ file-name-2 NEXT RECORD AT END	4. Following the random READ in INX-TEST-10.01, the next three records are read sequentially and each record check. This test expects records to retrieved in the sequence defined by the ALTERNATE KEY (data-name-5) values e.g., record sequence order 51, 49 and 48. See VI - 4.4.4(15) The READ statement.
011		OBJECTIVE: Use the random READ statement to establish the current record pointer for

subsequent sequential READs. The random READ uses the optional KEY IS phrase in its construct.

011.01 READ file-name-2
NEXT RECORD
AT END

1. The file is opened as input to establish the current record pointer to the first record currently existing within the file and establish the prime record key (data-name-4) as the key of reference. The READ statement is executed and the record is verified as being the first record associated with the prime record key.

011.02 READ file-name-2
RECORD KEY IS
data-name-5
INVALID KEY

2. The alternate record key (data-name-5) is loaded with the key value to retrieve the 1st record as defined by the alternate key (the 300th record created). The read statement is executed and the record contents verified.

See VI - 4.4.4(15) The READ statement.

011.03 READ file-name-2
NEXT RECORD AT
END

3. The file is read until 200 records have been processed. Each record is verified for accuracy. With the DYNAMIC mode specified, it is expected that the key of reference used for this READ statement (format 1) will be the same key of reference used by the format 2 READ statement above.

See VI - 4.4.4(15) The READ statement.

011.04 READ file-name-2
KEY IS EQUAL
data-name-4

READ file-name-2
NEXT RECORD
AT END

4. Verify that the key of reference established by the Format 2 READ statement in 2 above and used by the Format 1 READ statement in 3 above can be reestablished to a different key of reference. The READ statement is used to establish the prime record key as key of reference. The file is then read sequentially until 200 records have been processed. It is expected that the records will be read in sequence as defined by the prime record key (data-name-4).

See VI - 4.4.4(15) The READ statement.

012 OBJECTIVE: Verify that when the Format 2 READ statement is used without the KEY phrase, the key of reference established is the prime record key (data-name-4) and the same one used for subsequent executions of Format 1 READ statements until a different key of reference is established.

See VI - 4.4.4(16).

012.01	OPEN INPUT file-name-2 START file-name-2 KEY IS EQUAL TO data-name-5 READ file-name-2 NEXT RECORD AT END	1. Establish the alternate key (data-name-5) as the key of reference and verify that the records can be accessed by that key. The alternate record key (data-name-5) is loaded with the key value to retrieve the 1st record as defined by the alternate key. The START statement establishes the key of reference and 100 records are read. The records are verified to assure correct record retrieval.
012.02	READ file-name-2	2. Use the READ statement without the KEY IS and INVALID KEY phrase to establish the prime record key as the key of reference. The prime record key (data-name-4) is loaded with the key value of the 100th record. Following the read, the record is checked to verify that the correct record was retrieved. Record number 100 is expected to be retrieved.

See VI - 4.4.4(16) The READ statement.

012.03	READ file-name-2 NEXT RECORD AT END	3. The Format 1 READ statement is used to retrieve the records sequentially. When the file is in the DYNAMIC mode the key of reference should be that established by the random READ statement above. One hundred records are read and the contents checked to verify that the correct records were retrieved.

See VI - 4.4.4(16) The READ statement.

IX211 (Name of run unit)

GENERAL: The function of this run unit is to test the permissible language syntax and language semantic relating to non-unique RECORD KEY data-items. Qualification is required to uniquely identify the RECORD KEY when referenced explicitly. The OPEN and START statements are tested for correct positioning of the current record pointer for non-unique RECORD KEY data-items. The elements tested in this run unit are:

(1) READ Statement requiring qualification in the KEY phrase;
(2) START Statement requiring qualification in the KEY phrase;
(3) START Statement without explicit key qualification
(4) RECORD KEY requiring qualification.
(5) OPEN statement used to position the current record pointer for a non-unique RECORD KEY.

Each element tested will be exercised semantically by this routine.

The appropriate implementor-names which are to be supplied for this program are identified by x-cards. The x-cards used by this program are:

x-24	Indexed file-1 for first ASSIGN TO clause
x-25	Indexed file-2 for second ASSIGN TO clause
x-26	Indexed file-3 for third ASSIGN TO clause
x-55	System printer
x-74	VALUE OF implementor-name
x-75	Object of VALUE OF for file-1
x-76	Object of VALUE OF for file-2
x-77	Object of VALUE OF for file-3
x-82	Source-Computer
x-83	Object-Computer

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	OBJECTIVE: Create and verify 3 files which require qualification to uniquely identify the RECORD KEY. File-name-1 and file-name-2 require qualification to uniquely identify the RECORD KEY between files in the program. File-name-3 requires qualification to uniquely identify a RECORD KEY data-item within a record associated with the file.	OBJECTIVE: Create three files for testing the READ, OPEN and START statements. The characteristics defined in the SELECT clause are used for files file-name-1, file-name-2 and file-name-3 respectively. Each file will contain the following file characteristics: File size = 300 records Record size = 240 characters Blocking = 1 record RECORD KEY = 10 characters RECORD KEY sequence = increasing numerical value by increments of 2 (2 thru 600) for file-name-1, increasing numerical value by increments of 2 (302 thru 900) for file-name-2 and increasing numerical value by increments of 1 (1 thru 300) for file-name-3. (NOTE: For file-name-3 data-items, other than the one associated with the RECORD KEY, which have the same reference name (data-name-2), will be 10 positions in length and contain a decreasing numerical value by increments of 1 (300 thru 1).
001.01	SELECT file-name-1 ORGANIZATION IS INDEXED ACCESS MODE IS DYNAMIC RECORD KEY IS data-name-1 OF record-name-1 See VI - 2.1.2.2 DATA DIVISION: The RECORD KEY (data-name-1) has the same name as that specified for file-name-2. See VI - 2.1.2.3 WRITE record-name-1 INVALID KEY	1. This test creates a file containing 300 records. Following creation of the file, a check is made to see that the WRITE was executed 300 times and that there were no INVALID KEY paths taken.

001.02 SELECT file-name-2
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS data-name-1
OF record-name-2

See VI - 2.1.2.2

DATA DIVISION: The RECORD KEY is the same name as that specified for file-name-1.

See VI - 2.1.2.3

WRITE record-name-2 INVALID KEY

1. This test creates a file containing 300 records. Following creation of the file a check is made to see that the WRITE was executed 300 times and that there were no INVALID KEY paths taken.

001.03 SELECT file-name-3
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS data-name-2
OF data-name-3
IN data-name-4

See VI - 2.1.2.2

DATA DIVISION: The data description for file-name-3 contains multiple data-name-2s and data-name-3s, however, is uniquely identified by the next level of qualification i.e., data-name-4.

WRITE record-name-3 INVALID KEY

3. This test creates a file containing 300 records. Following creation of the file a check is made to see that the WRITE statement was executed 300 times and that there were no INVALID KEY paths taken.

002 OBJECTIVE: Use the OPEN statement to to establish the RECORD KEY as the key of reference for subsequent READs. The OPEN statement should position the current record to the first record in the file as defined by the RECORD KEY for that file. Any qualification necessary to establish the appropriate RECORD KEY for future record retrieval is expected to be handled by the implementor.

002.01 OPEN INPUT file-name-1

1. File-name-1 is opened as INPUT.

	READ file-name-1 NEXT RECORD AT END	Ten records of file-name-1 are read sequentially and their contents verified for accurracy. The first 10 records of file-name-1 are expected to be retrieved.
002.02	OPEN I-O file-name-2 READ file-name-2 NEXT RECORD AT END	2. File-name-2 are opened as I-O. Ten records of file-name-2 are read sequentially and their contents verified for accuracy. The first 10 records of file-name-2 are expected to be retrieved.
002.03	OPEN INPUT file-name-3 READ file-name-3 NEXT RECORD AT END	3. File-name-3 is opened as INPUT and 10 records are READ sequentially. The first 10 records of file-name-3 as defined by RECORD KEY contents associated with data-name-2 OF data-name-3 IN data-name-4, are expected to be retrieved. The contents of each record are verified for accuracy and a check made to see that there was not a premature AT END path taken while reading the file.
003	OBJECTIVE: Use of qualification to uniquely reference a RECORD KEY via use of a READ statement.	OBJECTIVE: Retrieve records from a file via the READ in which non-unique RECORD KEY names have been used. Qualification is used where necessary to uniquely identify the RECORD KEY.
003.01	READ file-name-3 RECORD KEY IS data-name-2 OF data-name-3 IN data-name-4 INVALID KEY (NOTE: Qualification is required to uniquely identify a data-item within a record of that file. See VI - 4.4.3(3)	1. Open file-name-3 as I-O and load the RECORD KEY (data-name-2 OF data-name-3 IN data-name-4) with a predetermined record key value. The READ is executed and the contents of the record are verified . A check is made to see that an INVALID KEY path was not taken.
003.02	READ file-name-1 RECORD KEY IS data-name-1 OF record-name-1 INVALID KEY (NOTE: Qualification is used to identify a record key associated with a particular file) See VI - 4.4.3(3)	2. Load the RECORD KEY data-item for the file with a predetermined key value The record is retrieved randomly, the record verified and a check is made to see that the INVALID KEY path was not taken.
003.03	READ file-name-2 RECORD INVALID KEY	3 Load the RECORD KEY data-item for the file with a predetermined key value.

	(NOTE: The RECORD KEY data-item used for this READ is expected to be the one associated with file-name-2.)	The record is retrieved randomly, the record verified and a check made to to see that the INVALID KEY path was not taken.
004	OBJECTIVE: Use of qualification to uniquely reference a RECORD KEY via use of a START Statement.	OBJECTIVE: Postion the current record pointer for the file in which non-unique RECORD KEY names are used. Where necessary, qualification is used in the START statement to uniquely identify the RECORD KEY data-item.
004.01	START file-name-3 KEY IS EQUAL TO data-name-2 IN data-name-3 OF data-name-4 INVALID KEY (NOTE: Qualification is required to uniquely identify a data-item within a record associated with file-name-3.) See VI - 4.6.3 (3)	1. Oper file-name-3 as I-O and load the RECORD KEY data-item with a predetermined value. Following exexution of the START, the file is read sequentially, the record verified and a check made to see that the INVALID KEY path was not taken.
004.02	START file-name-2 KEY IS EQUAL TO data-name-1 OF record-name-2 INVALID KEY (NOTE: qualification used to identify a record key associated with a particular file) See VI - 4.6.3(3)	2. Open file-name-2 as I-O and load the RECORD KEY data-item with a predetermined value. Following execution of the START, the file is read sequentially, the record verified and a check is made to see that the INVALID KEY path was not taken.
004.03	START file-name-1 INVALID KEY (NOTE: The RECORD KEY data-item (data-name 1) used for the START is expected to be one associated with file-name-1) See VI - 4.6.3 (3)	3. Open file-name-1 as I-O and load the RECORD KEY data-item with a predetermined value. Following execution of the START, the file is read sequentially, the record verified and a check is made to see that the INVALID KEY path was not taken.

IX212 (Name of run unit)

GENERAL: The function of this run unit is to test the permissible syntactical constructs of COBOL elements associated with level 2 of INDEXED I-O. The elements tested in this routine are:

(1) READ Statement (format 2);
(2) START Statement;
(3) ALTERNATE RECORD KEYS series;
(4) ALTERNATE RECORD KEY requiring qualification.

Each element tested will be exercised semantically by this routine.

X-Card parameters which must be supplied for this program are:

X-24	Indexed file-1 for ASSIGN TO clause
X-25	Indexed file-2 for ASSIGN TO clause
X-55	System printer
X-74	VALUE OF implementor-name
X-75	Object of VALUE OF clause for file X-24 above
X-76	Object of VALUE OF clause for file X-25 above
X-82	Source-computer
X-83	Object-computer

TEST INX-TEST	SYNTACTICAL CONSTRUCT	SEMANTIC ACTION
001	OBJECTIVE: Create and verify a file which contains multiple ALTERNATE RECORD KEYs. One ALTERNATE KEY requires qualification to distinguish it between files within the program.	
001.01	SELECT file-name-1 ORGANIZATION IS INDEXED ACCESS MODE IS DYNAMIC RECORD KEY IS data-name-1 ALTERNATE RECORD KEY IS data-name-2 ALTERNATE RECORD KEY IS data-name-3 WITH DUPLICATES ALTERNATE RECORD KEY IS data-name-4 OF record-name-1 See VI - 2.1.2.2 DATA DIVISION: The ALTERNATE RECORD KEY (data-name-4) is the same data-name as that specified for file-name-2. however. is uniquely qualified by record-name-1. See VI - 2.1.2.3 WRITE record-name-1 INVALID KEY	1. This test creates a file containing 300 records. The RECORD KEY (data-name-1) will be 6 positions in length and established sequentially. The ALTERNATE KEY (data-name-2) will be 12 positions in length and established in inverse sequence. The ALTERNATE KEY (data-name-3) will be 18 positions in length and established in sequence by a logical incremental value of 5 with every 50th and 51st record a duplicate Key. The ALTERNATE KEY (data-name-4) will be 24 positions in length and establish in sequence by a logical incremental value of 10. The keys will be created in a work area and stored into the record as a group item.
001.02		2. OPEN file-name-1 as I-O. Establish data-name-2 as the key of reference via the START statement. Read 10 records of file-name-1 and verify that the file can be accessed via the ALTERNATE KEY (data-name-2).
001.03		3. Read 60 records of file-name-1 using the ALTERNATE KEY (data-name-3) and

verify that logical records 100 and 101 contain duplicate Keys and can be accessed. The START Statement is used to establish the Key of reference. The Key record pointer is established as the 41st logical record of the file.

002 OBJECTIVE: Create and verify a file which contains multiple ALTERNATE RECORD KEYS. One ALTERNATE KEY requires qualification to distinguish it between data-items within the record description of the file.

002.01

```
SELECT file-name-2
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS data-name-5
ALTERNATE RECORD KEY IS
    data-name-6 OF data-name-8
ALTERNATE RECORD KEY IS
    data-name-7 WITH
    DUPLICATES
ALTERNATE RECORD KEY IS
    data-name-4   OF
    record-name-2
```

See VI - 2.1.2.2

DATA DIVISION: The data description for file-name-2 contains multiple data-name-6s which can be uniquely identified through qualification (data-name-8). The ALTERNATE RECORD KEY (data-name-4) is the same data-name as that specified for file-name-1. however, is uniquely qualified by record-name-2.

See VI - 2.1.2.3

```
WRITE record-name-2
      INVALID KEY
```

1. This test creates a file containing 300 records. The RECORD KEY (data-name-5) will be 6 positions in length and established sequentially. The ALTERNATE KEY (data-name-4) will be 12 positions in length and established in inverse sequence. The ALTERNATE KEY (data-name-7) will be 18 positions in length and established in sequence by a logical incremental value of 5 with every 50th and 51st record a duplicate key. The ALTERNATE KEY (data-name-6) will be 24 positions in length and establish in sequence by a logical incremental value of 10. The keys will be created in a work area and stored into the record as a group item.

002.02		2. OPEN file-name-2 as I-O. Establish data-name-6 OF data-name-8 as the key of reference via the START statement. Read 10 records of file-name-1 sequentially and verify that the file can be accessed via the ALTEPNATE KEY (data-name-6). 3. Read 60 records of file-name-2 sequentially using the ALTERNATE KEY (data-name-7) and verify that logical records 100 and 101 which contain duplicated keys, can be accessed. The START statement is used to establish the key of reference and position the record pointer to the 41st logical record of the file.
003	OBJECTIVE: Use of qualification to uniquely reference an ALTERNATE KEY via use of a READ statement.	
003.01	READ file-name-2 KEY IS data-name-6 OF data-name-8 (OBJECTIVE: Qualification used within a record of the file being tested) See VI - 4.4.3(3)	1. Open file-name-2 as I-O and load the ALTERNATE KEY (data-name-6) with a predetermined record key value. The record is retrieved randomly and verified.
003.02	READ file-name-1 KEY IS data-name-4 IN record-name-1 (OBJECTIVE: Qualification used to identify a record key associated with a particular file) See VI - 4.4.3(3)	2. Load the ALTEPNATE KEY (data-name-4) for file-name-1 with a predetermined record key value. The record is retrieved randomly and verified.
004	OBJECTIVE: Use of qualification to uniquely reference an ALTEPNATE KEY via use of a START Statement.	
004.01	START file-name-2 KEY IS EQUAL TO data-name-6 IN data-name-8 (Note qualification used within a record of the file being tested)	1. Open file-name-2 as I-O and establish key of reference via the START Statement. The record is then read sequentially via the "READ file-name-2 NEXT RECORD" Statement and verified.

004.02	START file-name-2 KEY IS EQUAL TO data-name-4 OF record-name IN file-name-2 (Note qualification used to identify a record key associated with a particular file.) See VI - 4.6.3(3)	2. Open file-name-2 as I-O and establish key or reference via the START Statement. The record is then read sequentially via the "READ file-name-2 NEXT RECORD" Statement and verified.

IX213 (Name of run unit)

GENERAL: The function of this run unit is to test the language semantics relating to duplicate key values and appropriate positioning of the current record pointer. For any key value which is the same as one already existing in the file for a prime record key or an alternate record key in which the DUPLICATES phrase has not been specified should cause the appropriate INVALID KEY path to be taken and the FILE STATUS data-item to be updated. This run unit tests the correct functioning of the START statement for its respective INVALID KEY paths, establishing the key of reference for sequential reads and using a data-item other than the RECORD KEY data-name to position the current record pointer. The elements tested in this run unit are:

(1) FILE STATUS
(2) WRITE INVALID KEY (duplicate key values)
(3) START statement

The appropriate implementor-names which are to be supplied for this program are identified by X-cards. The X-cards used by this program are:

X-24 Indexed file-1 for ASSIGN TO clause
X-55 System printer
X-74 VALUE OF implementor name
X-75 Object of VALUE OF for file-1
X-82 Source-Computer
X-83 Object-Computer

TEST INX-TEST	CONSTRUCT	SEMANTIC ACTIONS
001	SELECT file-name-1 ORGANIZETION IS INDEX RECORD KEY IS data-name-1 ALTERNATE KEY IS data-name-4 ALTERNATE KEY IS data-name-10 WITH DUPLICATES FILE STATUS IS data-name-16 ACCESS MODE IS SEQUENTIAL	OBJECTIVE: Create an Indexed file using the SEQUENTIAL ACCESS MODE. The records will contain 2 alternate keys, one with duplicates keys and the other without. The file contains 200 records. The WRITE statement will be executed more than 200 times expecting the INVALID KEY path to be taken and the FILE STATUS data-item to be updated for any key in which a duplicated key value is not permitted. Any key value provided which is not unique for a prime record key or for an alternate record key in which the DUPLICATES phrase has not been specified should not permit the record to be written to the file.
	USE AFTER STANDARD ERROR PROCEDURE ON file-name-1	The file will have the following characteristics:
	WRITE record-name INVALID KEY	File size = 201 records Record size = 240 characters RECORD KEY = 15 characters data-name-1 data-name-2 X(10) values B thru U data-name-3 X(5) values 0 thru 400 by increments of 2) (sample of key contents) BBBBBBBBBB000 BBBBBBBBBC002 BBBBBBBBCC004 CCCCCCCCCC020 CCCCCCCCCD021 UUUUUUUUUU400 ALTERNATE KEY = 20 characters data-name-4 data-name-5

```
                data-name-6    X(5) values E thru Y
                data-name-7    X(5) values E thru Y
              data-name-8      X(3) values 0 thru
                                    400 by increments
                                    of 2
            data-name-9        X(7) value "ALTKEY1"

              (sample of key contents)

                  EEEEEEEEE000ALTKEY1
                  EEEEEEEEEF002ALTKEY1
                  EEEEEEEEFF004ALTKEY1


                  FFFFFFFFFF020ALTKEY1
                  FFFFFFFFFG022ALTKEY1


                  YYYYYYYYYY400ALTKEY1

ALTERNATE KEY      = 20 characters
  data-name-10
    data-name-11
      data-name-12  X(5) values W thru D
      data-name-13  X(5) values W thru D
    data-name-14    X(3) values 400 thru 0
                         by decrements of 2

    data-name-15    X(7)  value "ALTKEY2"

      (sample of key contents)

          WWWWWWWWWW400ALTKEY2
          WWWWWWWWWV398ALTKEY2
          WWWWWWWWVV396ALTKEY2
          WWWWWWWVVV394ALTKEY2


          WVVVVVVVVV382ALTKEY2
          VVVVVVVVVV380ALTKEY2
          VVVVVVVVVV380ALTKEY2
          VVVVVVVVU378ALTKEY2


          DDDDDDDDDD000ALTKEY2

NOTE:  After the first record, data-name-10
       will be duplicated every 10th
       record i.e., 10th and 11th record
       will have the same value.
```

The Key contents are given in the sequence in which they are created. The record retrieval sequence is expected to be in ascending order of record Key values therefore, records accessed by the alternate Key (data-name-10) would cause records to be retrieved in descending order of RECORD KEY value (data-name-1).

001.01 WRITE record-name INVALID KEY

1. Following the 201st execution of the WRITE statement a test is made to verify that there were no unexpected INVALID KEY paths taken. All records thru the 201st record are released in ascending order of prime record Key values. The alternate Key values are not released in ascending order.

See VI - 4.8.4 (12) The WRITE statement (level 1)

001.02

2. The 202nd time the WRITE is executed the RECORD KEY (data-name-1) is loaded with a Key value the same as one already existing in the file. The INVALID KEY path is expected to be taken. The FILE STATUS (data-name-16) is saved for future reference.

See VI - 4.8.4 (10), (15) The WRITE STATEMENT (level 1)

001.03

3. The 203rd time the WRITE is executed the ALTERNATE KEY (data-name-4) is loaded with a Key value the same as one already existing in the file. The value for the RECORD KEY (data-name-1) is loaded with a unique value. Since the DUPLICATES phrase was not specified, the INVALID KEY path of the WRITE statement is expected to be taken. The FILE STATUS (data-name-16) is saved for future reference.

00'.04

4. The 204th time the WRITE is executed RECORD KEY (data-name-1) is loaded with a unique Key value which has a lower sequential Key value than the previous record. When the SEQUENTIAL ACCESS MODE has been specified, any such condition should cause an INVALID KEY condition to occur. The FILE STATUS data-item

(data-name-16) is saved for future reference.

See VI - 4.8.4 (15)a The WRITE statment (level 1)

001.05

5. The WRITE in INX-TEST-001.02 above should cause the FILE STATUS data-item to be updated as well. The data-item (data-name-16) should reflect that the record contained a RECORD KEY value already present in the file. Thus data-name-16 should contain the value "22".

See VI - 4.8.4 (15)a The WRITE statement (level 1)
1.3.4 I-O Status

001.06

6. The WRITE in INX-TEST-001.03 above should have caused the FILE STATUS data-item (data-name-16) to be updated. The contents of data-name-16 should reflect that the record contained a ALTERNATE KEY (data-name-4) value already present in the file. Thus, contents of data-name-16 should contain the value "22".

See VI - 4.8.4 (15)c The WRITE statement 1.3.4 I-O Status.

001.07

7. The WRITE in INX-TEST-001.04 above should have caused the FILE STATUS data-item (data-name-16) to updated. The contents of data-name-16 should reflect that a record being written to the file contained a lower sequential value in the prime record key than previous record. Thus the contents of data-name-16 should contain the value "21".

See VI - 4.8.4 (15)a The WRITE statement (level 1)
1.3.4 I-O Status

002

OBJECTIVE: Read the file updated in INX-TEST-001 above and verify the file for accuracy. There is expected to be no more and no less than 201 records in the file. When an INVALID KEY condition

		occurs, as was expected for WRITEs 202 thru 204 in INX-TEST-001 above, execution of the input-output statement should have been unsuccessful and the file should not have been affected. This test checks to see that none of these records were written to the file. See 4.8.4 (16) The WRITE Statement
002.01	READ file-name-1 AT END	1. The file is opened as INPUT and read until the end-of-file is reached. A count is made as to the number of times the READ is executed. The READ should have executed 202 times. The 202nd time should have caused the AT END path to be taken.
002.02		2. During the read of the file in INX-TEST-002.01 each record is verified and a count kept of those records which did not match the expected record contents. This count is expected to be zero.
003	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the RECORD KEY (data-name-1).	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the RECORD KEY clause for key comparsions. If the KEY phrase of the START statement is not specified, the relational operator "IS EQUAL TO " is implied otherwise the relational operator in the KEY phrase is used. For this test, the relational operator associated with "EQUAL TO" comparsion will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
003.01	START file-name-1 INVALID KEY	1. Data-name-1 is loaded with a key value "[illegible]" which is a value not currently existing in the file. The key value given has a sequential location between two currently existing RECORD KEY values. The implied relational operator "IS EQUAL TO " is expected to be used for the compar-

ison thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (2), (3) and (6) The START statement.

003.02 START file-name-1
IS EQUAL TO
data-name-1
INVALID KEY

2. Data-name-1 is loaded with a Key value "BBBBBBBBBBB001" which is a value not currently existing in the file. The key value given has a sequential location between two currently existing RECORD KEY values. The explicit relation operator "IS EQUAL TO" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

003.03 START file-name-1
KEY IS EQUAL TO
data-name-1
INVALID KEY

3. Data-name-1 is loaded with a Key value "BBBBBBBBBA000" which is a value with a magitude of one less then the first RECORD KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparison thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

003.04 START file-name-1
KEY IS EQUAL TO
data-name-1
INVALID KEY

4. Data-name-1 is loaded with a Key value "UUUUUUUUUU401" which is a value with a magitude of one more than the last RECORD KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparsion thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

003.05 START file-name-1
KEY IS EQUAL TO
data-name-2
INVALID KEY

See VI - 4.6.3 (5)

5. Data-name-2 is loaded with a value "CCCCCCD022". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. if truncation occurs on the left a Key match will be found thus an INVALID KEY condition

will not occur. For this test an INVALID KEY condition is expected.

See VI - 4.6.4 (3) The START statement.

004	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the RECORD KEY (data-name-1). See test 004.05 below. See VI - 4.6.3 (5)	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the RECORD KEY for comparisons. For this test the relational operator associated with the "GREATER THAN" comparison will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
004.01	START file-name-1 KEY IS > data-name-1 INVALID KEY	1. Data-name-1 is loaded with a key value "VVVVVVVVV000" which is a value greater than any currently in the file. The symbol for "GREATER THAN" is used to designate the comparison. The relational operator associated with the "GREATER THAN " comparison is expected to be used thus, the record should not be found which should cause an INVALID KEY condition to occur. See VI - 4.6.4 (2), (3) and (6) The START statement.
004.02	START file-name-1 KEY IS GREATER THAN data-name-1 INVALID KEY	2. Data-name-1 is loaded with a key value "UUUUUUUUUU500" which is a value greater than any currently in the file. The relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
004.03	START file-name-1 KEY IS GREATER THAN data-name-1 INVALID KEY	3. Data-name-1 is loaded with a key value "UUUUUUUUUU400" which is a value for the last record in the file. The comparison "GREATER THAN" should not find a RECORD KEY value greater than this in the file. An INVALID KEY

		condition is expected to occur. See VI - 4.6.4 (3) and (5) The START statement.
004.04	START file-name-1 KEY IS GRAEATER THAN data-name-1 INVALID KEY	4. Data-name-1 is loaded with a Key value "UUUUUUUUUU401" which is a value with a magitude of one more than the last RECORD KEY value in the file. The explicit relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
004.05	START file-name-1 KEY IS GREATER THAN data-name-2 INVALID KEY See VI - 4.6.3 (5)	5. Data-name-2 is loaded with a value "UUUUUUUUUU". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. Since the first 10 positions of the RECORD KEY are equal to that of data-name-2 the last record Key value in the file should be equal. A value of one magnitude larger than this value should not be present in the file thus an INVALID KEY condition is expected. See VI - 4.6.4 (3) The START statement.
005	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the RECORD KEY (data-na - 1). See test 005.04 below. See VI - 4.6.3 (5)	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the RECORD KEY for KEY comparisons For this test the relational operator associated with the "NOT LESS THAN" comparison will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists. the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
005.01	START file-name-1 KEY IS NOT <	1. Data-name-1 is loaded with a Key value "VVVVVVVV000" which is a value

	data-name-1 INVALID KEY	greater than any currently in the file. The symbol for "LESS THAN " is used to designate the comparison. The relational operator associated with the "NOT LESS THAN" comparison is expected to be used for the comparison thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (2), (3) and (6) The START statement.
005.02	START file-name-1 KEY IS NOT LESS THAN data-name-1 INVALID KEY	2. Data-name-1 is loaded with a key value "UUUUUUUUUU500" which is a value greater than any currently in the file. The relational operator "NOT LESS THEN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
005.03	START file-name-1 KEY IS NOT LESS THAN data-name-1 INVALID KEY	3. Data-name-1 is loaded with a key value "UUUUUUUUUU401" which is a value of one larger than any existing in the file. The comparison "NOT LESS THAN" should not find a RECORD KEY value which matches. An INVALID KEY condition is expected to occur. See VI - 4.6.4 (3) and (5) The START statement.
005.04	START file-name-1 KEY IS NOT LESS THAN data-name-2 INVALID KEY See VI - 4.6.3 (5)	4. Data-name-2 is loaded with a value "UUUUUUUUUV". When the operands being compared are of unequal size comparison proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. The first 10 positions of the RECORD KEY are equal to that of data-name-2 and the last record key value in the file has a value of "UUUUUUUUUU" in the first 10 positions of the record key The specified value is one larger than any in the file. A RECORD KEY value equal to or greater than data-name-2 should not be found thus an INVALID KEY is expected. See VI - 4.6.4 (3) The START statement.

006	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-4). See test 006.05 below. See VI - 4.6.3 (5)	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the ALTERNATE KEY for KEY comparisons. The ALTERNATE KEY clause does not contain the DUPLICATES phrase. If the KEY phrase of the START statement is not specified, the relational operator "IS EQUAL TO" is implied otherwise the relational operator in the KEY phrase is used. For this test, the relational operator associated with "EQUAL TO" comparsion will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
006.01	START file-name-1 INVALID KEY	1. Data-name-4 is loaded with a Key value "FFFFFFFFFF021ALTKEY1" which is a value not currently existing in the file. The key value given has a sequential location between two currently existing ALTERNATE KEY values. Data-name-1 is loaded with the key value "CCCCCCCCCC020" which is a RECORD KEY for the logical record immediately preceeding that specified for the ALTERNATE KEY (data-name-4). If the KEY phrase is not specified the START uses the RECORD KEY and "IS EQUAL TO" for the comparisons thus an INVALID KEY condition is not expected. See VI - 4.6.4 (2), (3) and (6) The START statement.
006.02	START file-name-1 KEY IS EQUAL TO data-name-4 INVALID KEY	2. Data-name-4 is loaded with a Key value "EEEEEEEEEE001ALTKEY1" which is a value not currently existing in the file. The key value given has a sequential location between two currently existing ALTERNATE KEY values. An explicit relation operator "IS EQUAL TO" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START

		statement.
006.03	START file-name-1 KEY IS EQUAL TO data-name-4 INVALID KEY	3. Data-name-4 is loaded with a key value "EEEEEEEEED000ALTKEY1" which is a value with a magitude of one less then the first ALTERNATE KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparison thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
006.04	START file-name-1 KEY IS EQUAL TO data-name-4 INVALID KEY	4. Data-name-4 is loaded with a key value "YYYYYYYYYY401ALTKEY1" which is a value with a magitude of one more than the last ALTERNATE KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
006.05	START file-name-1 KEY IS EQUAL TO data-name-5 INVALID KEY See VI - 4.6.3 (5)	5. Data-name-5 is loaded with a value "022ALTKEY1". The operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. If truncation occurs on the left a key match will be found thus an INVALID KEY condition will not occur. For this test an INVALID KEY condition is expected. See VI - 4.6.4 (3) The START statement.
007	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-4). See test 007.05 below. See VI - 4.5.3 (5)	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the ALTERNATE KEY for KEY comparisons. The ALTERNATE KEY clause does not contain the DUPLICATES option. For this test the relational operator associated with the "GREATER THAN" comparsion will be tested. If the comparison is not satisfied by any record in the file. an INVALID KEY condition exists. the START statement is

unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.

007.01 START file-name-1
KEY IS >
data-name-4
INVALID KEY

1. Data-name-4 is loaded with a Key value "ZZZZZZZZZZ401ALTKEY1" which is a value greater than any currently in the file. The symbol for "GREATER THAN" is used to designate the comparison. The relational operator associated with the "GREATER THAN " comparison is expected to be used. The record should not be found thus cause an INVALID KEY to occur.

See VI - 4.6.4 (2), (3) and (6) The START statement.

007.02 START file-name-1
KEY IS GREATER THAN
data-name-4
INVALID KEY

2. Data-name-4 is loaded with a Key value "ZZZZZZZZZZ400ALTKEY1" which is a value greater than any currently in the file. The relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

007.03 START file-name-1
KEY IS GREATER THAN
data-name-4
INVALID KEY

3. Data-name-4 is loaded with a Key value "YYYYYYYYYY400ALTKEY" which is the value for the last record in the file. The comparison "GREATER THAN" should not find a ALTERNATE KEY value which matches. An INVALID KEY condition is expected to occur.

See VI - 4.6.4 (3) and (5) The START statement.

007.04 START file-name-1
KEY IS GREATER THAN
data-name-4
INVALID KEY

4. Data-name-4 is loaded with a Key value "YYYYYYYYYY401ALTKEY1" which is a value with a magnitude of one more than the last ALTERNATE KEY value in the file. The explicit relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

007.05	START file-name-1 KEY IS GREATER THAN data-name-5 INVALID KEY See VI - 4.6.3 (5)	5. Data-name-5 is loaded with a value "YYYYYYYYYY". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. Since the first 10 positions of the ALTERNATE KEY are equal to that of data-name-5 the last record key value in the file should be equal. A value of one magnitude larger than the value specified by data-name-2 should not be present in the file thus an INVALID KEY condition is expected. See VI - 4.6.4 (3) The START statement.
008	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-4).	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-item referenced in the ALTERNATE KEY clause associated with the file for comparsions. The ALTERNATE KEY clause does not contain the DUPLICATES option. For this test the relational operator associated with the "NOT LESS THAN" comparison will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
008.01	START file-name-1 KEY IS NOT < data-name-4 INVALID KEY	1. Data-name-4 is loaded with a Key value "ZZZZZZZZZZ000ALTKEY1" which is a value greater than any currently in the file. The symbol for "LESS THAN " is used to designate the comparison. The relational operator associated with the "NOT LESS THAN" comparison is expected to be used for the comparison thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (2), (3) and (6) The START statement.

008.02	START file-name-1 KEY IS NOT LESS THAN data-name-4 INVALID KEY	2. Data-name-4 is loaded with a key value "YYYYYYYYY500ALTKEY1" which is a value greater than any currently in the file. The relational operator "NOT LESS THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
008.03	START file-name-1 KEY IS NOT LESS THAN data-name-4 INVALID KEY	3. Data-name-4 is loaded with a key value "YYYYYYYYY401ALTKEY1" which is a value of one larger than any existing in the file. The comparison "NOT LESS THAN" should not find a ALTERNATE KEY value which matches. An INVALID KEY condition is expected to occur. See VI - 4.6.4 (3) and (5) The START statement.
008.04	START file-name-1 KEY IS NOT LESS THAN data-name-2 INVALID KEY See VI - 4.6.3 (5)	4. Data-name-5 is loaded with a value "YYYYYYYYYZ". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. The first 10 positions of the ALTERNATE KEY are equal to that of data-name-2, and the last record key value in the file has a value of "YYYYYYYYYY" in the first 10 positions of the record key. The specified value is one larger than any in the file. A ALTERNATE KEY value equal to or greater than data-name-2 should not be found thus an INVALID KEY is expected. See VI - 4.6.4 (3) The START statement.
009	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-10).	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the ALTERNATE KEY for KEY comparisons. The ALTERNATE KEY clause contains the DUPLICATES option. If the KEY phrase of the START statement is not specified, the relational operator "IS EQUAL TO" is implied otherwise the relational operator in the KEY phrase

is used. For this test, the relational operator associated with "EQUAL TO" comparsion wi'l be tested.
If the comparison is not satisfied by any rerord in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.

009.01 START file-name-1
INVALID KEY

1. Data-name-10 is loaded with a Key value "WWWWWWWVVV395ALTKEY2" which is a value not currently existing in the file. The Key value given has a sequential location between two currently existing ALTERNATE KEY values. The implied relational operator "IS EQUAL TO " is expected to be used for the comparison thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (2), (3) and (6) The START statement.

009.02 START file-name-1
KEY IS EQUAL TO
data-name-4
INVALID KEY

2. Data-name-10 is loaded with a Key value "WWWWWWWWW381ALTKEY2" which is a value not currently existing in the file. The Key value given has a sequential location between two currently existing ALTERNATE KEY values and one greater than a Key value which is a duplicate of another in the file. The explicit relation operator "IS EQUAL TO" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

006.03 START file-name-1
KEY IS EQUAL TO
data-name-10
INVALID KEY

3. Data-name-10 is loaded with a key value "DDDDDDDDDC000ALTKEY2" which is a value with a magnitude of one less than the first ALTERNATE KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparison thus cause an INVALID KEY condition to occur.

See VI - 4.6.4 (3) and (5) The START statement.

009.04	START file-name-1 KEY IS EQUAL TO data-name-10 INVALID KEY	4. Data-name-10 is loaded with a Key value "XXXXXXXXXX401ALTKEY2" which is a value with a magnitude of one more than the last ALTERNATE KEY value in the file. The explicit relational operator "IS EQUAL TO " is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
009.05	START file-name-1 KEY IS EQUAL TO data-name-11 INVALID KEY See VI - 4.6.3 (5)	5. Data-name-11 is loaded with a value "378ALTKEY2". When the operands being compared are of unequal size comparison proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. If truncation occurs on the left a key match will be found thus an INVALID KEY condition will not occur. For this test an INVALID KEY condition is expected. See VI - 4.6.4 (3) The START statement.
010	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-10).	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items asocciated with the ALTERNATE KEY for KEY comparisons. The ALTERNATE KEY clause contains the the DUPLICATES option. For this test the relational operator associated with the "GREATER THAN" comparsion will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
010.01	START file-name-1 KEY IS > data-name-10 INVALID KEY	1. Data-name-10 is loaded with a key value "ZZZZZZZZZZ401ALTKEY2" which is a value greater than any currently in the file. The symbol for "GREATER THAN" is used to designate the comparison. The relational operator associated with the "GREATER THAN " comparison

		is expected to he used for the comparison thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (2), (3) and (6) The START statement.
010.02	START file-name-1 KEY IS GREATER THAN data-name-10 INVALID KEY	2. Data-name-10 is loaded with a Key value "ZZZZZZZZZZ953ALTKEY2" which is a value greater than any currently in the file. The relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
010.03	START file-name-1 KEY IS GREATER THAN data-name-10 INVALID KEY	3. Data-name-10 is loaded with a Key value "WWWWWWWWW400ALTKEY2" which is the value for the last record in the file. The comparison "GREATER THAN" should not find a ALTERNATE KEY value which matches. An INVALID KEY condition is expected to occur. See VI - 4.6.4 (3) and (5) The START statement.
010.04	START file-name-1 KEY IS GREATER THAN data-name-10 INVALID KEY	4. Data-name-10 is loaded with a Key value "WWWWWWWWW401ALTKEY2" which is a value with a magnitude of one more than the last ALTERNATE KEY value in the file. The explicit relational operator "GREATER THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
010.05	START file-name-1 KEY IS GREATER THAN data-name-11 INVALID KEY See VI - 4.6.3 (5)	5. Data-name-11 is loaded with a value "WWWWWWWWW". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. Since the first 10 positions of the ALTERNATE KEY are equal to that of data-name-11 the last record key value in the file should be equal. A value of one magnitude larger than the value

		specified by data-name-11 should not be present in the file thus an INVALID KEY condition is expected. See VI - 4.6.4 (3) The START statement.
011	OBJECTIVE: The START statement is used specifying a data-item in the KEY phrase which is subordinate to the ALTERNATE KEY (data-name-10).	OBJECTIVE: Test the conditions which should cause the INVALID KEY path of the START statement to be taken. This test uses the data-items associated with the ALTERNATE KEY for KEY comparisons. The ALTERNATE KEY clause contains the DUPLICATES option. For this test the relational operator associated with the "NOT LESS THAN" comparison will be tested. If the comparison is not satisfied by any record in the file, an INVALID KEY condition exists, the START statement is unsuccessful and position of the current record pointer is undefined. For each execution of the START statement the contents of the FILE STATUS (data-name-16) is saved for future reference.
011.01	START file-name-1 KEY IS NOT < data-name-10 INVALID KEY	1. Data-name-10 is loaded with a key value "ZZZZZZZZZZ999ALTKEY2" which is a value greater than any currently in the file. The symbol for "LESS THAN " is used to designate the comparison. The relational operator associated with the "NOT LESS THAN" comparison is expected to be used for the comparison thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (2), (3) and (6) The START statement.
011.02	START file-name-1 KEY IS NOT LESS THAN data-name-10 INVALID KEY	2. Data-name-10 is loaded with a key value "WWWWWWWWW500ALTKEY2" which is a value greater than any currently in the file. The relational operator "NOT LESS THAN" is expected to be used for the comparsion thus cause an INVALID KEY condition to occur. See VI - 4.6.4 (3) and (5) The START statement.
011.03	START file-name-1 KEY IS NOT LESS THAN data-name-10	3. Data-name-10 is loaded with a key value "WWWWWWWWW401ALTKEY1" which is a value of one larger than any existing in the

	INVALID KEY	file. The comparison "NOT LESS THAN" should not find a ALTERNATE KEY value which matches. An INVALID KEY condition is expected to occur. See VI - 4.6.4 (3) and (5) The START statement.
011.04	START file-name-1 KEY IS NOT LESS THAN data-name-11 INVALID KEY See VI - 4.6.3 (5)	4. Data-name-11 is loaded with a value "WWWWWWWWWX". When the operands being compared are of unequal size comparsion proceeds as though the longer one were truncated on the right such that its length is equal to that of the shorter. The first 10 positions of the ALTERNATE KEY are equal to that of data-name-11 and the last record key value in the file has a value of "WWWWWWWWWW" in the first 10 positions of the record key. The specified value is one larger than any in the file. A ALTERNATE KEY value equal to or greater than data-name-10 should not be found thus an INVALID KEY is expected. See VI - 4.6.4 (3) The START statement.
012		OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-003 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.
012.01		1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-003.01. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".
012.02		2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-003.02. The key value specified for the START

should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

012.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-003.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

012.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-003.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

012.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-003.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

013

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-004 above.
The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

013.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-004.01. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

013.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-004.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

013.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-004.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

013.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-004.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

013.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-004.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

014

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-005 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

014.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-005.01. The key value specified for the START

should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

014.02 2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-005.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

014.03 3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-005.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

014.04 4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-005.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

015 OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-006 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

012.01 1. An INVALID KEY condition is not expected as a result of execution of the START statement in test INX-TEST-006.01. The key value specified for the START should find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "00".

015.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-006.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

015.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-006.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

015.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-006.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

015.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-006.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

016

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-007 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

016.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-007.01. The key value specified for the START

should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the valte "23".

016.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-007.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

016.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-007.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

016.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-007.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

016.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-007.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

017

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-008 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-1 is not expected to contain the correct file status code.

01/.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-008.01. The Key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

017.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-008.02. The Key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

017.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-008.03. The Key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

017.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-008.04. The Key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

018

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-009 above.
The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

018.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-009.01. The Key value specified for the START

should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

018.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-009.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

018.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-009.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a cona.tion and contain the value "23".

018.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-009.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

018.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-009.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

019

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-010 above. The following tests are dependent on the appropriate execution of the START statements i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

019.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-010.01. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

019.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-010.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

019.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-010.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

019.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-010.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

019.05

5. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-010.05. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

020

OBJECTIVE: This test checks the captured status contents of the FILE STATUS (data-name-16) from the START statements in test INX-TEST-011 above. The following tests are dependent on the appropriate execution of the START statements

i.e. successful or unsuccessful as the case may be. Thus if the START statement did not execute as expected the contents of data-name-16 is not expected to contain the correct file status code.

020.01

1. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-011.01. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

020.02

2. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-011.02. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

020.03

3. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-011.03. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".

020.04

4. An INVALID KEY condition is expected as a result of execution of the START statement in test INX-TEST-011.04. The key value specified for the START should not find a matching record in the file thus the FILE STATUS (data-name-16) should reflect such a condition and contain the value "23".